

FflasFfpack

Generated on Mon Nov 13 2023 00:00:00 for FflasFfpack by Doxygen 1.9.8

Mon Nov 13 2023 00:00:00

1 FFLAS-FFPACK Documentation.	1
1.1 Introduction	1
1.2 Goals	1
1.3 Design	1
1.4 Using FFLAS-FFPACK.	1
1.5 Contributing to fflas-ffpack, getting assistance.	2
1.6 Copying and Licence	2
1.7 Tutorial	2
1.8 Configuring and Installing FFLAS-FFPACK	2
1.9 Architecture of the library.	2
2 Bug List	3
3 Bibliography	7
4 Todo List	9
5 Topic Index	13
5.1 Topics	13
6 Namespace Index	15
6.1 Namespace List	15
7 Hierarchical Index	17
7.1 Class Hierarchy	17
8 Data Structure Index	25
8.1 Data Structures	25
9 File Index	33
9.1 File List	33
10 Topic Documentation	41
10.1 CHECKER	41
10.2 FFLAS-FFPACK	41
10.2.1 Detailed Description	41
10.2.2 FFLAS	42
10.2.3 Interfaces	42
10.3 Matrix Multiplication Algorithms	42
10.3.1 Detailed Description	42
10.4 SIMD wrapper	42
10.5 FFPACK	43
10.5.1 Detailed Description	43
10.6 FFLAS-FFPACK fields	43
10.6.1 Detailed Description	43
10.7 RNS	43

11 Namespace Documentation	45
11.1 FFLAS Namespace Reference	45
11.1.1 Typedef Documentation	72
11.1.1.1 Checker_fgemm	72
11.1.1.2 Checker_ftrsm	72
11.1.1.3 ForceCheck_fgemm	72
11.1.1.4 ForceCheck_ftrsm	72
11.1.1.5 ZOSparseMatrix	72
11.1.1.6 NotZOSparseMatrix	72
11.1.1.7 SimdSparseMatrix	72
11.1.1.8 NoSimdSparseMatrix	73
11.1.1.9 MKLSparseMatrixFormat	73
11.1.1.10 NotMKLSparseMatrixFormat	73
11.1.1.11 has_plus	73
11.1.1.12 has_minus	73
11.1.1.13 has_equal	73
11.1.1.14 has_plus_eq	73
11.1.1.15 has_minus_eq	73
11.1.1.16 has_mul	74
11.1.1.17 has_mul_eq	74
11.1.1.18 Timer	74
11.1.1.19 BaseTimer	74
11.1.1.20 UserTimer	74
11.1.1.21 SysTimer	74
11.1.2 Enumeration Type Documentation	74
11.1.2.1 FFLAS_ORDER	74
11.1.2.2 FFLAS_TRANSPOSE	75
11.1.2.3 FFLAS_UPLO	75
11.1.2.4 FFLAS_DIAG	75
11.1.2.5 FFLAS_SIDE	75
11.1.2.6 FFLAS_BASE	76
11.1.2.7 number_kind	76
11.1.2.8 SparseMatrix_t	76
11.1.2.9 FFLAS_FORMAT	77
11.1.3 Function Documentation	77
11.1.3.1 InfNorm()	77
11.1.3.2 min3()	77
11.1.3.3 max3()	77
11.1.3.4 min4()	77
11.1.3.5 max4()	78
11.1.3.6 fadd() [1/8]	78
11.1.3.7 faddin() [1/5]	78

11.1.3.8 fsub() [1/4]	78
11.1.3.9 fsubin() [1/3]	78
11.1.3.10 fadd() [2/8]	79
11.1.3.11 pfadd()	79
11.1.3.12 pfsub()	79
11.1.3.13 pfaddin()	80
11.1.3.14 pfsubin()	80
11.1.3.15 fadd() [3/8]	80
11.1.3.16 fsub() [2/4]	81
11.1.3.17 faddin() [2/5]	81
11.1.3.18 faddin() [3/5]	81
11.1.3.19 fsubin() [2/3]	82
11.1.3.20 fadd() [4/8]	82
11.1.3.21 fassign() [1/10]	83
11.1.3.22 fassign() [2/10]	83
11.1.3.23 fassign() [3/10]	83
11.1.3.24 fassign() [4/10]	84
11.1.3.25 fassign() [5/10]	84
11.1.3.26 fassign() [6/10]	84
11.1.3.27 fassign() [7/10]	84
11.1.3.28 fassign() [8/10]	85
11.1.3.29 faxpy() [1/6]	86
11.1.3.30 faxpy() [2/6]	86
11.1.3.31 faxpy() [3/6]	86
11.1.3.32 faxpy() [4/6]	87
11.1.3.33 fdot() [1/11]	87
11.1.3.34 fdot() [2/11]	87
11.1.3.35 fdot() [3/11]	88
11.1.3.36 fdot() [4/11]	88
11.1.3.37 fdot() [5/11]	88
11.1.3.38 fdot() [6/11]	88
11.1.3.39 fdot() [7/11]	89
11.1.3.40 fdot() [8/11]	89
11.1.3.41 fgemm() [1/23]	89
11.1.3.42 fgemm() [2/23]	90
11.1.3.43 fgemm() [3/23]	90
11.1.3.44 fgemm() [4/23]	90
11.1.3.45 fgemm() [5/23]	91
11.1.3.46 fgemm() [6/23]	92
11.1.3.47 fsquare() [1/6]	92
11.1.3.48 fsquare() [2/6]	93
11.1.3.49 fsquare() [3/6]	93

11.1.3.50 fsquare() [4/6]	93
11.1.3.51 fsquare() [5/6]	93
11.1.3.52 fgemm() [7/23]	94
11.1.3.53 fgemm() [8/23]	94
11.1.3.54 fgemm() [9/23]	94
11.1.3.55 fgemm() [10/23]	95
11.1.3.56 fgemm() [11/23]	95
11.1.3.57 fgemm() [12/23]	96
11.1.3.58 fgemm() [13/23]	96
11.1.3.59 fgemm() [14/23]	96
11.1.3.60 fgemm() [15/23]	97
11.1.3.61 fgemm() [16/23]	97
11.1.3.62 fgemm() [17/23]	97
11.1.3.63 fgemm() [18/23]	98
11.1.3.64 fgemv() [1/19]	98
11.1.3.65 fgemv() [2/19]	99
11.1.3.66 fgemv() [3/19]	99
11.1.3.67 fgemv() [4/19]	99
11.1.3.68 fgemv() [5/19]	100
11.1.3.69 fgemv() [6/19]	100
11.1.3.70 fgemv() [7/19]	101
11.1.3.71 fgemv() [8/19]	101
11.1.3.72 fgemv() [9/19]	101
11.1.3.73 fgemv() [10/19]	102
11.1.3.74 fgemv() [11/19]	102
11.1.3.75 fgemv() [12/19]	102
11.1.3.76 fgemv() [13/19]	103
11.1.3.77 fgemv() [14/19]	103
11.1.3.78 fgemv() [15/19]	103
11.1.3.79 fgemv() [16/19]	104
11.1.3.80 fger() [1/12]	104
11.1.3.81 fger() [2/12]	105
11.1.3.82 fger() [3/12]	105
11.1.3.83 fger() [4/12]	105
11.1.3.84 fger() [5/12]	106
11.1.3.85 fger() [6/12]	106
11.1.3.86 fger() [7/12]	106
11.1.3.87 fger() [8/12]	107
11.1.3.88 fger() [9/12]	107
11.1.3.89 fger() [10/12]	107
11.1.3.90 fger() [11/12]	108
11.1.3.91 freduce() [1/11]	108

11.1.3.92 <code>freduce()</code> [2/11]	108
11.1.3.93 <code>freduce_constoverride()</code> [1/2]	109
11.1.3.94 <code>finit()</code> [1/8]	109
11.1.3.95 <code>finit()</code> [2/8]	109
11.1.3.96 <code>freduce()</code> [3/11]	110
11.1.3.97 <code>freduce()</code> [4/11]	110
11.1.3.98 <code>pfreduce()</code>	110
11.1.3.99 <code>freduce()</code> [5/11]	110
11.1.3.100 <code>freduce_constoverride()</code> [2/2]	112
11.1.3.101 <code>finit()</code> [3/8]	112
11.1.3.102 <code>finit()</code> [4/8]	112
11.1.3.103 <code>freduce()</code> [6/11]	113
11.1.3.104 <code>freduce()</code> [7/11]	113
11.1.3.105 <code>freivalds()</code>	113
11.1.3.106 <code>fscalin()</code> [1/10]	114
11.1.3.107 <code>fscal()</code> [1/10]	114
11.1.3.108 <code>fscal()</code> [2/10]	115
11.1.3.109 <code>fscal()</code> [3/10]	115
11.1.3.110 <code>fscalin()</code> [2/10]	115
11.1.3.111 <code>fscalin()</code> [3/10]	116
11.1.3.112 <code>fscalin()</code> [4/10]	116
11.1.3.113 <code>fscal()</code> [4/10]	116
11.1.3.114 <code>fscalin()</code> [5/10]	117
11.1.3.115 <code>fscal()</code> [5/10]	117
11.1.3.116 <code>fscalin()</code> [6/10]	117
11.1.3.117 <code>fscal()</code> [6/10]	117
11.1.3.118 <code>fscalin()</code> [7/10]	118
11.1.3.119 <code>fscal()</code> [7/10]	118
11.1.3.120 <code>fscalin()</code> [8/10]	118
11.1.3.121 <code>fscal()</code> [8/10]	118
11.1.3.122 <code>fsyr2k()</code>	119
11.1.3.123 <code>fsyrk()</code> [1/16]	119
11.1.3.124 <code>fsyrk()</code> [2/16]	120
11.1.3.125 <code>fsyrk()</code> [3/16]	120
11.1.3.126 <code>fsyrk()</code> [4/16]	121
11.1.3.127 <code>fsyrk()</code> [5/16]	121
11.1.3.128 <code>fsyrk()</code> [6/16]	121
11.1.3.129 <code>fsyrk()</code> [7/16]	122
11.1.3.130 <code>fsyrk()</code> [8/16]	122
11.1.3.131 <code>fsyrk()</code> [9/16]	122
11.1.3.132 <code>fsyrk()</code> [10/16]	123
11.1.3.133 <code>fsyrk()</code> [11/16]	123

11.1.3.134 fsyrk() [12/16]	124
11.1.3.135 fsyrk() [13/16]	124
11.1.3.136 fsyrk() [14/16]	125
11.1.3.137 computeS1S2()	126
11.1.3.138 fsyrk() [15/16]	126
11.1.3.139 fsyrk() [16/16]	126
11.1.3.140 fsyrk_strassen() [1/2]	127
11.1.3.141 ftrmm() [1/3]	127
11.1.3.142 ftrmm() [2/3]	128
11.1.3.143 ftrsm() [1/9]	128
11.1.3.144 ftrsm() [2/9]	129
11.1.3.145 ftrsm() [3/9]	129
11.1.3.146 ftrsm() [4/9]	129
11.1.3.147 ftrsm() [5/9]	130
11.1.3.148 cblas_imptrsm()	130
11.1.3.149 ftrsv() [1/2]	130
11.1.3.150 igemm_()	131
11.1.3.151 finit() [5/8]	131
11.1.3.152 fconvert() [1/3]	132
11.1.3.153 fnegin() [1/4]	132
11.1.3.154 fneg() [1/4]	133
11.1.3.155 fzero() [1/5]	133
11.1.3.156 frand() [1/2]	133
11.1.3.157 fiszero() [1/4]	134
11.1.3.158 fequal() [1/4]	134
11.1.3.159 faxpby() [1/2]	135
11.1.3.160 fdot() [9/11]	135
11.1.3.161 fswap() [1/2]	135
11.1.3.162 fzero() [2/5]	137
11.1.3.163 fzero() [3/5]	137
11.1.3.164 frand() [2/2]	138
11.1.3.165 fequal() [2/4]	138
11.1.3.166 fiszero() [2/4]	139
11.1.3.167 fidentity() [1/4]	139
11.1.3.168 fidentity() [2/4]	139
11.1.3.169 finit() [6/8]	139
11.1.3.170 fconvert() [2/3]	140
11.1.3.171 fnegin() [2/4]	140
11.1.3.172 fneg() [2/4]	141
11.1.3.173 faxpby() [2/2]	141
11.1.3.174 fmove() [1/2]	142
11.1.3.175 bitsize()	142

11.1.3.176 <code>bitsize< Givaro::ZRing< Givaro::Integer > >()</code>	143
11.1.3.177 <code>ftmv()</code>	143
11.1.3.178 <code>ftsm()</code> [6/9]	143
11.1.3.179 <code>fsyrk_strassen()</code> [2/2]	144
11.1.3.180 <code>pfgemm()</code> [1/7]	144
11.1.3.181 <code>pfgemm_1D_rec()</code>	145
11.1.3.182 <code>pfgemm_2D_rec()</code>	145
11.1.3.183 <code>pfgemm_3D_rec()</code>	146
11.1.3.184 <code>pfgemm_3D_rec2()</code>	146
11.1.3.185 <code>fgemm()</code> [19/23]	146
11.1.3.186 <code>ftsm()</code> [7/9]	147
11.1.3.187 <code>ftsm()</code> [8/9]	147
11.1.3.188 <code>fspmv()</code> [1/2]	147
11.1.3.189 <code>fspmm()</code>	148
11.1.3.190 <code>sparse_init()</code> [1/16]	148
11.1.3.191 <code>sparse_init()</code> [2/16]	148
11.1.3.192 <code>sparse_delete()</code> [1/12]	148
11.1.3.193 <code>sparse_delete()</code> [2/12]	148
11.1.3.194 <code>sparse_init()</code> [3/16]	149
11.1.3.195 <code>sparse_init()</code> [4/16]	149
11.1.3.196 <code>sparse_delete()</code> [3/12]	149
11.1.3.197 <code>sparse_delete()</code> [4/12]	149
11.1.3.198 <code>sparse_print()</code> [1/3]	149
11.1.3.199 <code>sparse_init()</code> [5/16]	150
11.1.3.200 <code>sparse_init()</code> [6/16]	150
11.1.3.201 <code>sparse_init()</code> [7/16]	150
11.1.3.202 <code>sparse_init()</code> [8/16]	150
11.1.3.203 <code>sparse_delete()</code> [5/12]	151
11.1.3.204 <code>sparse_init()</code> [9/16]	151
11.1.3.205 <code>sparse_init()</code> [10/16]	151
11.1.3.206 <code>sparse_init()</code> [11/16]	151
11.1.3.207 <code>sparse_delete()</code> [6/12]	151
11.1.3.208 <code>sparse_delete()</code> [7/12]	152
11.1.3.209 <code>sparse_init()</code> [12/16]	152
11.1.3.210 <code>sparse_init()</code> [13/16]	152
11.1.3.211 <code>sparse_delete()</code> [8/12]	152
11.1.3.212 <code>sparse_delete()</code> [9/12]	152
11.1.3.213 <code>sparse_print()</code> [2/3]	152
11.1.3.214 <code>sparse_delete()</code> [10/12]	153
11.1.3.215 <code>sparse_init()</code> [14/16]	153
11.1.3.216 <code>operator<<()</code>	153
11.1.3.217 <code>readSmsFormat()</code>	153

11.1.3.218 readSprFormat()	153
11.1.3.219 getDataType() [1/4]	154
11.1.3.220 getDataType() [2/4]	154
11.1.3.221 getDataType() [3/4]	154
11.1.3.222 getDataType() [4/4]	154
11.1.3.223 readMachineType()	154
11.1.3.224 readDnsFormat()	154
11.1.3.225 writeDnsFormat()	154
11.1.3.226 fspmv() [2/2]	155
11.1.3.227 sparse_delete() [11/12]	155
11.1.3.228 sparse_delete() [12/12]	155
11.1.3.229 sparse_print() [3/3]	155
11.1.3.230 sparse_init() [15/16]	155
11.1.3.231 sparse_init() [16/16]	155
11.1.3.232 computeDeviation()	156
11.1.3.233 getStat()	156
11.1.3.234 maxCardinality()	156
11.1.3.235 maxCardinality< Givaro::Modular< int64_t > >()	156
11.1.3.236 maxCardinality< Givaro::Modular< int32_t > >()	156
11.1.3.237 minCardinality()	156
11.1.3.238 fflas_delete() [1/4]	156
11.1.3.239 fflas_delete() [2/4]	157
11.1.3.240 fflas_new() [1/7]	157
11.1.3.241 fflas_new() [2/7]	157
11.1.3.242 finit_rns() [1/2]	157
11.1.3.243 finit_trans_rns()	157
11.1.3.244 fconvert_rns() [1/2]	158
11.1.3.245 fconvert_trans_rns()	158
11.1.3.246 fflas_new() [3/7]	158
11.1.3.247 fflas_new() [4/7]	158
11.1.3.248 finit_rns() [2/2]	158
11.1.3.249 fconvert_rns() [2/2]	159
11.1.3.250 freduce() [8/11]	159
11.1.3.251 freduce() [9/11]	159
11.1.3.252 finit() [7/8]	160
11.1.3.253 fconvert() [3/3]	160
11.1.3.254 fnegin() [3/4]	161
11.1.3.255 fneg() [3/4]	161
11.1.3.256 fzero() [4/5]	161
11.1.3.257 fiszero() [3/4]	162
11.1.3.258 fequal() [3/4]	162
11.1.3.259 fassign() [9/10]	163

11.1.3.260 fscaln()	[9/10]	163
11.1.3.261 fscal()	[9/10]	164
11.1.3.262 faxpy()	[5/6]	164
11.1.3.263 fdot()	[10/11]	165
11.1.3.264 fswap()	[2/2]	165
11.1.3.265 fadd()	[5/8]	166
11.1.3.266 fsub()	[3/4]	166
11.1.3.267 faddin()	[4/5]	166
11.1.3.268 fadd()	[6/8]	166
11.1.3.269 fassign()	[10/10]	167
11.1.3.270 fzero()	[5/5]	167
11.1.3.271 fequal()	[4/4]	167
11.1.3.272 fiszero()	[4/4]	168
11.1.3.273 fidentity()	[3/4]	168
11.1.3.274 fidentity()	[4/4]	169
11.1.3.275 freduce()	[10/11]	169
11.1.3.276 freduce()	[11/11]	169
11.1.3.277 finit()	[8/8]	170
11.1.3.278 fnegin()	[4/4]	170
11.1.3.279 fneg()	[4/4]	170
11.1.3.280 fscaln()	[10/10]	171
11.1.3.281 fscal()	[10/10]	171
11.1.3.282 faxpy()	[6/6]	172
11.1.3.283 fmove()	[2/2]	172
11.1.3.284 fadd()	[7/8]	173
11.1.3.285 fsub()	[4/4]	174
11.1.3.286 fsubin()	[3/3]	174
11.1.3.287 fadd()	[8/8]	175
11.1.3.288 faddin()	[5/5]	175
11.1.3.289 fgemv()	[17/19]	175
11.1.3.290 fger()	[12/12]	176
11.1.3.291 ftrsv()	[2/2]	177
11.1.3.292 ftrsm()	[9/9]	177
11.1.3.293 ftrmm()	[3/3]	178
11.1.3.294 fgemm()	[20/23]	179
11.1.3.295 fgemm()	[21/23]	180
11.1.3.296 fgemm()	[22/23]	180
11.1.3.297 fgemm()	[23/23]	180
11.1.3.298 fsquare()	[6/6]	181
11.1.3.299 BlockCuts()	[1/2]	181
11.1.3.300 BlockCuts< CuttingStrategy::Single, StrategyParameter::Threads >()		181
11.1.3.301 BlockCuts< CuttingStrategy::Row, StrategyParameter::Fixed >()		182

11.1.3.302 BlockCuts< CuttingStrategy::Row, StrategyParameter::Grain >()	182
11.1.3.303 BlockCuts< CuttingStrategy::Block, StrategyParameter::Grain >()	182
11.1.3.304 BlockCuts< CuttingStrategy::Column, StrategyParameter::Fixed >()	182
11.1.3.305 BlockCuts< CuttingStrategy::Column, StrategyParameter::Grain >()	183
11.1.3.306 BlockCuts< CuttingStrategy::Block, StrategyParameter::Fixed >()	183
11.1.3.307 BlockCuts< CuttingStrategy::Row, StrategyParameter::Threads >()	183
11.1.3.308 BlockCuts< CuttingStrategy::Column, StrategyParameter::Threads >()	183
11.1.3.309 BlockCuts< CuttingStrategy::Block, StrategyParameter::Threads >()	183
11.1.3.310 BlockCuts() [2/2]	184
11.1.3.311 pfzero()	184
11.1.3.312 pfrand()	184
11.1.3.313 fdot() [11/11]	184
11.1.3.314 pfgemm() [2/7]	185
11.1.3.315 pfgemm() [3/7]	185
11.1.3.316 pfgemm() [4/7]	185
11.1.3.317 pfgemm() [5/7]	186
11.1.3.318 pfgemm() [6/7]	186
11.1.3.319 pfgemm() [7/7]	187
11.1.3.320 fgemv() [18/19]	187
11.1.3.321 fgemv() [19/19]	187
11.1.3.322 parseArguments()	188
11.1.3.323 getArgumentValue()	188
11.1.3.324 writeCommandString()	188
11.1.3.325 WriteMatrix() [1/2]	188
11.1.3.326 preamble()	190
11.1.3.327 ReadMatrix() [1/2]	190
11.1.3.328 ReadMatrix() [2/2]	190
11.1.3.329 WriteMatrix() [2/2]	191
11.1.3.330 WritePermutation()	191
11.1.3.331 alignable()	192
11.1.3.332 alignable< Givaro::Integer * >()	192
11.1.3.333 fflas_new() [5/7]	192
11.1.3.334 fflas_new() [6/7]	192
11.1.3.335 fflas_new() [7/7]	192
11.1.3.336 fflas_delete() [3/4]	192
11.1.3.337 fflas_delete() [4/4]	192
11.1.3.338 prefetch()	193
11.1.3.339 getTLBSize()	193
11.1.3.340 queryCacheSizes()	193
11.1.3.341 queryL1CacheSize()	193
11.1.3.342 queryTopLevelCacheSize()	193
11.1.3.343 getSeed()	193

11.2 FFLAS::_ftranspose_impl Namespace Reference	194
11.2.1 Function Documentation	194
11.2.1.1 not_inplace()	194
11.2.1.2 square_inplace()	194
11.2.1.3 nonsquare_inplace_v1()	194
11.2.1.4 nonsquare_inplace_v2()	195
11.3 FFLAS::BLAS3 Namespace Reference	195
11.3.1 Function Documentation	196
11.3.1.1 Bini()	196
11.3.1.2 WinoPar()	197
11.3.1.3 Winograd()	197
11.3.1.4 WinogradAcc_3_23()	197
11.3.1.5 WinogradAcc_3_21()	198
11.3.1.6 WinogradAcc_2_24()	198
11.3.1.7 WinogradAcc_2_27()	199
11.3.1.8 WinogradAcc_LR()	199
11.3.1.9 WinogradAcc_R_S()	199
11.3.1.10 WinogradAcc_L_S()	200
11.3.1.11 Winograd_LR_S()	200
11.3.1.12 Winograd_L_S()	200
11.3.1.13 Winograd_R_S()	201
11.4 FFLAS::csr_hyb_details Namespace Reference	201
11.5 FFLAS::CuttingStrategy Namespace Reference	201
11.5.1 Typedef Documentation	202
11.5.1.1 RNSModulus	202
11.6 FFLAS::details Namespace Reference	202
11.6.1 Function Documentation	203
11.6.1.1 fadd() [1/5]	203
11.6.1.2 fadd() [2/5]	204
11.6.1.3 fadd() [3/5]	204
11.6.1.4 fadd() [4/5]	204
11.6.1.5 fadd() [5/5]	205
11.6.1.6 faxpy() [1/2]	205
11.6.1.7 faxpy() [2/2]	205
11.6.1.8 freduce() [1/4]	205
11.6.1.9 freduce() [2/4]	206
11.6.1.10 freduce() [3/4]	206
11.6.1.11 freduce() [4/4]	206
11.6.1.12 fscaln() [1/2]	206
11.6.1.13 fscal() [1/2]	207
11.6.1.14 fscaln() [2/2]	207
11.6.1.15 fscal() [2/2]	207

11.6.1.16 igebb44()	207
11.6.1.17 igebb24()	208
11.6.1.18 igebb14()	208
11.6.1.19 igebb41()	208
11.6.1.20 igebb21()	208
11.6.1.21 igebb11()	209
11.6.1.22 igebp()	209
11.6.1.23 pack_lhs()	209
11.6.1.24 pack_rhs()	210
11.6.1.25 gebp()	210
11.6.1.26 BlockingFactor()	210
11.7 FFLAS::details_spmv Namespace Reference	210
11.8 FFLAS::ElementCategories Namespace Reference	211
11.9 FFLAS::FieldCategories Namespace Reference	211
11.9.1 Detailed Description	211
11.10 FFLAS::MMHelperAlgo Namespace Reference	211
11.11 FFLAS::ModeCategories Namespace Reference	212
11.11.1 Detailed Description	212
11.12 FFLAS::ParSeqHelper Namespace Reference	212
11.12.1 Detailed Description	212
11.13 FFLAS::Protected Namespace Reference	213
11.13.1 Function Documentation	216
11.13.1.1 computeFactorClassic() [1/3]	216
11.13.1.2 computeFactorClassic() [2/3]	216
11.13.1.3 computeFactorClassic() [3/3]	216
11.13.1.4 DotProdBoundClassic()	216
11.13.1.5 TRSMBound() [1/3]	217
11.13.1.6 TRSMBound() [2/3]	217
11.13.1.7 TRSMBound() [3/3]	217
11.13.1.8 fgemm_convert()	218
11.13.1.9 NeedPreAddReduction() [1/2]	218
11.13.1.10 NeedPreAddReduction() [2/2]	218
11.13.1.11 NeedPreSubReduction() [1/2]	218
11.13.1.12 NeedPreSubReduction() [2/2]	219
11.13.1.13 NeedDoublePreAddReduction() [1/2]	219
11.13.1.14 NeedDoublePreAddReduction() [2/2]	219
11.13.1.15 ScalAndReduce() [1/3]	219
11.13.1.16 ScalAndReduce() [2/3]	220
11.13.1.17 fsquareCommon()	220
11.13.1.18 WinogradThreshold() [1/4]	220
11.13.1.19 WinogradThreshold() [2/4]	220
11.13.1.20 WinogradThreshold() [3/4]	220

11.13.1.21 WinogradThreshold() [4/4]	221
11.13.1.22 WinogradSteps()	221
11.13.1.23 DynamicPeeling()	221
11.13.1.24 DynamicPeeling2()	221
11.13.1.25 WinogradCalc()	222
11.13.1.26 fgemv_convert()	222
11.13.1.27 fger_convert()	223
11.13.1.28 fsyrk_convert()	223
11.13.1.29 ScalAndReduce() [3/3]	223
11.13.1.30 NeedPreScalReduction() [1/2]	224
11.13.1.31 NeedPreScalReduction() [2/2]	224
11.13.1.32 NeedPreAxyReduction() [1/2]	224
11.13.1.33 NeedPreAxyReduction() [2/2]	224
11.13.1.34 min_types() [1/7]	224
11.13.1.35 min_types() [2/7]	225
11.13.1.36 min_types() [3/7]	225
11.13.1.37 min_types() [4/7]	225
11.13.1.38 min_types() [5/7]	225
11.13.1.39 min_types() [6/7]	225
11.13.1.40 min_types() [7/7]	225
11.13.1.41 unfit() [1/4]	225
11.13.1.42 unfit() [2/4]	225
11.13.1.43 unfit() [3/4]	226
11.13.1.44 unfit() [4/4]	226
11.13.1.45 igemm_colmajor() [1/2]	226
11.13.1.46 igemm_colmajor() [2/2]	226
11.13.1.47 igemm()	226
11.13.1.48 MatF2MatD_Triangular()	227
11.13.1.49 MatF2MatFI_Triangular()	227
11.14 FFLAS::sell_details Namespace Reference	227
11.15 FFLAS::sparse_details Namespace Reference	228
11.15.1 Function Documentation	231
11.15.1.1 init_y() [1/2]	231
11.15.1.2 init_y() [2/2]	231
11.15.1.3 fspmv_dispatch() [1/2]	231
11.15.1.4 fspmv_dispatch() [2/2]	231
11.15.1.5 fspmv() [1/12]	232
11.15.1.6 fspmv() [2/12]	232
11.15.1.7 fspmv() [3/12]	232
11.15.1.8 fspmv() [4/12]	232
11.15.1.9 fspmv() [5/12]	232
11.15.1.10 fspmv() [6/12]	233

11.15.1.11 fspmv()	[7/12]	233
11.15.1.12 fspmv()	[8/12]	233
11.15.1.13 fspmv()	[9/12]	233
11.15.1.14 fspmm_dispatch()	[1/2]	234
11.15.1.15 fspmm_dispatch()	[2/2]	234
11.15.1.16 fspmm()	[1/9]	234
11.15.1.17 fspmm()	[2/9]	234
11.15.1.18 fspmm()	[3/9]	235
11.15.1.19 fspmm()	[4/9]	235
11.15.1.20 fspmm()	[5/9]	235
11.15.1.21 fspmm()	[6/9]	236
11.15.1.22 fspmm()	[7/9]	236
11.15.1.23 fspmm()	[8/9]	236
11.15.1.24 fspmm()	[9/9]	236
11.15.1.25 pfspmm_dispatch()	[1/2]	237
11.15.1.26 pfspmm_dispatch()	[2/2]	237
11.15.1.27 pfspmm()	[1/9]	237
11.15.1.28 pfspmm()	[2/9]	237
11.15.1.29 pfspmm()	[3/9]	238
11.15.1.30 pfspmm()	[4/9]	238
11.15.1.31 pfspmm()	[5/9]	238
11.15.1.32 pfspmm()	[6/9]	239
11.15.1.33 pfspmm()	[7/9]	239
11.15.1.34 pfspmm()	[8/9]	239
11.15.1.35 pfspmm()	[9/9]	239
11.15.1.36 pfspmv()	[1/6]	240
11.15.1.37 pfspmv()	[2/6]	240
11.15.1.38 pfspmv()	[3/6]	240
11.15.1.39 pfspmv()	[4/6]	240
11.15.1.40 pfspmv()	[5/6]	240
11.15.1.41 pfspmv()	[6/6]	241
11.15.1.42 fspmv()	[10/12]	241
11.15.1.43 fspmv()	[11/12]	241
11.15.1.44 fspmv()	[12/12]	241
11.16 FFLAS::sparse_details_impl Namespace Reference		242
11.16.1 Function Documentation		250
11.16.1.1 fspmm()	[1/15]	250
11.16.1.2 fspmm()	[2/15]	250
11.16.1.3 fspmm()	[3/15]	251
11.16.1.4 fspmm_simd_aligned()	[1/2]	251
11.16.1.5 fspmm_simd_unaligned()	[1/2]	251
11.16.1.6 fspmm_one()	[1/4]	251

11.16.1.7 fspmm_mone() [1/4]	252
11.16.1.8 fspmm_one_simd_aligned() [1/3]	252
11.16.1.9 fspmm_one_simd_unaligned() [1/3]	252
11.16.1.10 fspmm_mone_simd_aligned() [1/3]	252
11.16.1.11 fspmm_mone_simd_unaligned() [1/3]	253
11.16.1.12 fspmv() [1/21]	253
11.16.1.13 fspmv() [2/21]	253
11.16.1.14 fspmv() [3/21]	253
11.16.1.15 fspmv_one() [1/10]	253
11.16.1.16 fspmv_mone() [1/10]	254
11.16.1.17 fspmv_one() [2/10]	254
11.16.1.18 fspmv_mone() [2/10]	254
11.16.1.19 pfspmm() [1/18]	254
11.16.1.20 pfspmm() [2/18]	254
11.16.1.21 pfspmm() [3/18]	255
11.16.1.22 pfspmm_one() [1/2]	255
11.16.1.23 pfspmm_mone() [1/2]	255
11.16.1.24 pfspmm_one() [2/2]	255
11.16.1.25 pfspmm_mone() [2/2]	256
11.16.1.26 pfspmv() [1/18]	256
11.16.1.27 pfspmv_task()	256
11.16.1.28 pfspmv() [2/18]	256
11.16.1.29 pfspmv() [3/18]	256
11.16.1.30 pfspmv_one() [1/8]	257
11.16.1.31 pfspmv_mone() [1/8]	257
11.16.1.32 pfspmv_one() [2/8]	257
11.16.1.33 pfspmv_mone() [2/8]	257
11.16.1.34 fspmm() [4/15]	257
11.16.1.35 fspmm() [5/15]	258
11.16.1.36 fspmm_simd_aligned() [2/2]	258
11.16.1.37 fspmm_simd_unaligned() [2/2]	258
11.16.1.38 fspmm() [6/15]	258
11.16.1.39 fspmm_one() [2/4]	259
11.16.1.40 fspmm_mone() [2/4]	259
11.16.1.41 fspmm_one_simd_aligned() [2/3]	259
11.16.1.42 fspmm_one_simd_unaligned() [2/3]	259
11.16.1.43 fspmm_mone_simd_aligned() [2/3]	260
11.16.1.44 fspmm_mone_simd_unaligned() [2/3]	260
11.16.1.45 fspmv() [4/21]	260
11.16.1.46 fspmv() [5/21]	260
11.16.1.47 fspmv() [6/21]	260
11.16.1.48 fspmv_one() [3/10]	261

11.16.1.49 fspmv_mone()	[3/10]	261
11.16.1.50 fspmv_one()	[4/10]	261
11.16.1.51 fspmv_mone()	[4/10]	261
11.16.1.52 pfspmm()	[4/18]	261
11.16.1.53 pfspmm()	[5/18]	262
11.16.1.54 pfspmm()	[6/18]	262
11.16.1.55 pfspmm()	[7/18]	262
11.16.1.56 pfspmm()	[8/18]	262
11.16.1.57 pfspmm()	[9/18]	263
11.16.1.58 pfspmv()	[4/18]	263
11.16.1.59 pfspmv()	[5/18]	263
11.16.1.60 pfspmv()	[6/18]	263
11.16.1.61 fspmm()	[7/15]	263
11.16.1.62 fspmm()	[8/15]	264
11.16.1.63 fspmm()	[9/15]	264
11.16.1.64 fspmv()	[7/21]	264
11.16.1.65 fspmv()	[8/21]	264
11.16.1.66 fspmv()	[9/21]	264
11.16.1.67 pfspmm()	[10/18]	265
11.16.1.68 pfspmm()	[11/18]	265
11.16.1.69 pfspmm()	[12/18]	265
11.16.1.70 pfspmm()	[13/18]	265
11.16.1.71 pfspmm()	[14/18]	266
11.16.1.72 pfspmm()	[15/18]	266
11.16.1.73 pfspmm_zo()	[1/2]	266
11.16.1.74 pfspmm_zo()	[2/2]	266
11.16.1.75 pfspmv()	[7/18]	267
11.16.1.76 pfspmv()	[8/18]	267
11.16.1.77 pfspmv()	[9/18]	267
11.16.1.78 pfspmv_one()	[3/8]	267
11.16.1.79 pfspmv_mone()	[3/8]	267
11.16.1.80 pfspmv_one()	[4/8]	268
11.16.1.81 pfspmv_mone()	[4/8]	268
11.16.1.82 fspmm()	[10/15]	268
11.16.1.83 fspmm()	[11/15]	268
11.16.1.84 fspmm()	[12/15]	269
11.16.1.85 fspmm_mone()	[3/4]	269
11.16.1.86 fspmm_one()	[3/4]	269
11.16.1.87 fspmm_mone()	[4/4]	269
11.16.1.88 fspmm_one()	[4/4]	270
11.16.1.89 fspmm_one_simd_aligned()	[3/3]	270
11.16.1.90 fspmm_one_simd_unaligned()	[3/3]	270

11.16.1.91 fspmm_mone_simd_aligned() [3/3]	270
11.16.1.92 fspmm_mone_simd_unaligned() [3/3]	271
11.16.1.93 fspmv() [10/21]	271
11.16.1.94 fspmv() [11/21]	271
11.16.1.95 fspmv() [12/21]	271
11.16.1.96 fspmv_one() [5/10]	271
11.16.1.97 fspmv_mone() [5/10]	272
11.16.1.98 fspmv_one() [6/10]	272
11.16.1.99 fspmv_mone() [6/10]	272
11.16.1.100 pfspmv() [10/18]	272
11.16.1.101 pfspmv() [11/18]	272
11.16.1.102 pfspmv() [12/18]	273
11.16.1.103 pfspmv_one() [5/8]	273
11.16.1.104 pfspmv_mone() [5/8]	273
11.16.1.105 pfspmv_one() [6/8]	273
11.16.1.106 pfspmv_mone() [6/8]	273
11.16.1.107 fspmv() [13/21]	274
11.16.1.108 fspmv_simd() [1/4]	274
11.16.1.109 fspmv() [14/21]	274
11.16.1.110 fspmv_simd() [2/4]	274
11.16.1.111 fspmv() [15/21]	274
11.16.1.112 fspmv_one() [7/10]	275
11.16.1.113 fspmv_mone() [7/10]	275
11.16.1.114 fspmv_one() [8/10]	275
11.16.1.115 fspmv_mone() [8/10]	275
11.16.1.116 fspmv_one_simd() [1/2]	275
11.16.1.117 fspmv_mone_simd() [1/2]	276
11.16.1.118 pfspm() [16/18]	276
11.16.1.119 pfspm() [17/18]	276
11.16.1.120 pfspm() [18/18]	276
11.16.1.121 pfspmv() [13/18]	277
11.16.1.122 pfspmv() [14/18]	277
11.16.1.123 pfspmv() [15/18]	277
11.16.1.124 fspmm() [13/15]	277
11.16.1.125 fspmm() [14/15]	277
11.16.1.126 fspmm() [15/15]	278
11.16.1.127 fspmv() [16/21]	278
11.16.1.128 fspmv() [17/21]	278
11.16.1.129 fspmv() [18/21]	278
11.16.1.130 pfspmv() [16/18]	278
11.16.1.131 pfspmv() [17/18]	279
11.16.1.132 pfspmv() [18/18]	279

11.16.1.133 pfspmv_one() [7/8]	279
11.16.1.134 pfspmv_mone() [7/8]	279
11.16.1.135 pfspmv_one() [8/8]	279
11.16.1.136 pfspmv_mone() [8/8]	280
11.16.1.137 fspmv() [19/21]	280
11.16.1.138 fspmv_simd() [3/4]	280
11.16.1.139 fspmv() [20/21]	280
11.16.1.140 fspmv_simd() [4/4]	280
11.16.1.141 fspmv() [21/21]	281
11.16.1.142 fspmv_one() [9/10]	281
11.16.1.143 fspmv_mone() [9/10]	281
11.16.1.144 fspmv_one_simd() [2/2]	281
11.16.1.145 fspmv_mone_simd() [2/2]	281
11.16.1.146 fspmv_one() [10/10]	282
11.16.1.147 fspmv_mone() [10/10]	282
11.17 FFLAS::StrategyParameter Namespace Reference	282
11.18 FFLAS::StructureHelper Namespace Reference	282
11.18.1 Detailed Description	282
11.19 FFLAS::vectorised Namespace Reference	283
11.19.1 Function Documentation	284
11.19.1.1 VEC_ADD()	284
11.19.1.2 addp()	284
11.19.1.3 VEC_SUB()	285
11.19.1.4 subp()	285
11.19.1.5 add()	285
11.19.1.6 sub()	285
11.19.1.7 axpyp() [1/2]	285
11.19.1.8 axpyp() [2/2]	286
11.19.1.9 reduce() [1/9]	286
11.19.1.10 reduce() [2/9]	286
11.19.1.11 reduce() [3/9]	286
11.19.1.12 reduce() [4/9]	286
11.19.1.13 reduce() [5/9]	286
11.19.1.14 reduce() [6/9]	287
11.19.1.15 reduce() [7/9]	287
11.19.1.16 reduce() [8/9]	287
11.19.1.17 reduce() [9/9]	287
11.19.1.18 modp() [1/2]	287
11.19.1.19 modp() [2/2]	287
11.19.1.20 scalp() [1/3]	288
11.19.1.21 scalp() [2/3]	288
11.19.1.22 scalp() [3/3]	288

11.20 FFLAS::vectorised::unswitch Namespace Reference	288
11.20.1 Function Documentation	289
11.20.1.1 axpyp() [1/2]	289
11.20.1.2 axpyp() [2/2]	289
11.20.1.3 modp() [1/2]	290
11.20.1.4 modp() [2/2]	290
11.20.1.5 scalp() [1/3]	290
11.20.1.6 scalp() [2/3]	290
11.20.1.7 scalp() [3/3]	291
11.21 FFPACK Namespace Reference	291
11.21.1 Detailed Description	307
11.21.2 Typedef Documentation	307
11.21.2.1 Checker_PLUQ	307
11.21.2.2 Checker_Det	308
11.21.2.3 Checker_invert	308
11.21.2.4 Checker_charpoly	308
11.21.2.5 ForceCheck_PLUQ	308
11.21.2.6 ForceCheck_Det	308
11.21.2.7 ForceCheck_invert	308
11.21.2.8 ForceCheck_charpoly	308
11.21.3 Function Documentation	308
11.21.3.1 LAPACKPerm2MathPerm()	308
11.21.3.2 MathPerm2LAPACKPerm()	309
11.21.3.3 applyP() [1/4]	309
11.21.3.4 applyP() [2/4]	310
11.21.3.5 applyP() [3/4]	310
11.21.3.6 MonotonicApplyP()	310
11.21.3.7 fgetrs() [1/4]	311
11.21.3.8 fgetrs() [2/4]	312
11.21.3.9 fgesv() [1/4]	313
11.21.3.10 fgesv() [2/4]	313
11.21.3.11 ftrtri() [1/2]	314
11.21.3.12 trinv_left() [1/2]	314
11.21.3.13 ftrtrm() [1/2]	315
11.21.3.14 ftrstr()	315
11.21.3.15 ftrssyr2k()	316
11.21.3.16 fsytrf() [1/3]	316
11.21.3.17 fsytrf() [2/3]	317
11.21.3.18 fsytrf() [3/3]	317
11.21.3.19 fsytrf_nonunit() [1/3]	318
11.21.3.20 PLUQ() [1/6]	318
11.21.3.21 pPLUQ()	319

11.21.3.22 PLUQ() [2/6]	319
11.21.3.23 PLUQ() [3/6]	320
11.21.3.24 LUdivine() [1/4]	320
11.21.3.25 ColumnEchelonForm() [1/3]	321
11.21.3.26 pColumnEchelonForm()	321
11.21.3.27 ColumnEchelonForm() [2/3]	322
11.21.3.28 RowEchelonForm() [1/3]	322
11.21.3.29 pRowEchelonForm()	322
11.21.3.30 RowEchelonForm() [2/3]	323
11.21.3.31 ReducedColumnEchelonForm() [1/3]	323
11.21.3.32 pReducedColumnEchelonForm()	324
11.21.3.33 ReducedColumnEchelonForm() [2/3]	324
11.21.3.34 ReducedRowEchelonForm() [1/3]	324
11.21.3.35 pReducedRowEchelonForm()	325
11.21.3.36 ReducedRowEchelonForm() [2/3]	325
11.21.3.37 Invert() [1/4]	325
11.21.3.38 Invert() [2/4]	326
11.21.3.39 Invert2() [1/2]	326
11.21.3.40 CharPoly() [1/8]	327
11.21.3.41 CharPoly() [2/8]	328
11.21.3.42 CharPoly() [3/8]	328
11.21.3.43 MinPoly() [1/4]	329
11.21.3.44 MinPoly() [2/4]	329
11.21.3.45 MatVecMinPoly() [1/2]	330
11.21.3.46 Rank() [1/3]	330
11.21.3.47 pRank()	331
11.21.3.48 Rank() [2/3]	331
11.21.3.49 IsSingular() [1/2]	331
11.21.3.50 Det() [1/6]	332
11.21.3.51 pDet()	332
11.21.3.52 Det() [2/6]	333
11.21.3.53 Solve() [1/3]	333
11.21.3.54 Solve() [2/3]	333
11.21.3.55 pSolve()	334
11.21.3.56 RandomNullSpaceVector() [1/3]	334
11.21.3.57 NullSpaceBasis() [1/2]	335
11.21.3.58 RowRankProfile() [1/3]	335
11.21.3.59 pRowRankProfile()	336
11.21.3.60 RowRankProfile() [2/3]	336
11.21.3.61 ColumnRankProfile() [1/3]	336
11.21.3.62 pColumnRankProfile()	337
11.21.3.63 ColumnRankProfile() [2/3]	337

11.21.3.64 RankProfileFromLU()	337
11.21.3.65 LeadingSubmatrixRankProfiles()	338
11.21.3.66 RowRankProfileSubmatrixIndices() [1/2]	338
11.21.3.67 ColRankProfileSubmatrixIndices() [1/2]	339
11.21.3.68 RowRankProfileSubmatrix() [1/2]	340
11.21.3.69 ColRankProfileSubmatrix() [1/2]	340
11.21.3.70 getTriangular() [1/2]	341
11.21.3.71 getTriangular() [2/2]	342
11.21.3.72 getEchelonForm() [1/2]	342
11.21.3.73 getEchelonForm() [2/2]	343
11.21.3.74 getEchelonTransform()	344
11.21.3.75 getReducedEchelonForm() [1/2]	345
11.21.3.76 getReducedEchelonForm() [2/2]	345
11.21.3.77 getReducedEchelonTransform()	346
11.21.3.78 PLUQtoEchelonPermutation()	347
11.21.3.79 LTBruhatGen()	347
11.21.3.80 getLTBruhatGen() [1/2]	347
11.21.3.81 getLTBruhatGen() [2/2]	348
11.21.3.82 LTQSorder()	349
11.21.3.83 CompressToBlockBiDiagonal()	349
11.21.3.84 ExpandBlockBiDiagonalToBruhat()	350
11.21.3.85 Bruhat2EchelonPermutation()	350
11.21.3.86 TInverter() [1/2]	351
11.21.3.87 ComputeRPermutation() [1/2]	351
11.21.3.88 productBruhatxTS() [1/2]	351
11.21.3.89 LQUPtoInverseOfFullRankMinor() [1/2]	352
11.21.3.90 RandomNullSpaceVector() [2/3]	352
11.21.3.91 solveLB() [1/2]	353
11.21.3.92 solveLB2() [1/2]	353
11.21.3.93 TInverter() [2/2]	354
11.21.3.94 ComputeRPermutation() [2/2]	354
11.21.3.95 expandLCRE()	354
11.21.3.96 productBruhatxTS() [2/2]	354
11.21.3.97 Danilevski()	356
11.21.3.98 buildMatrix()	356
11.21.3.99 CharPoly() [4/8]	356
11.21.3.100 CharPoly() [5/8]	356
11.21.3.101 Det() [3/6]	357
11.21.3.102 Det() [4/6]	357
11.21.3.103 fsytrf_BC_Crout()	357
11.21.3.104 fsytrf_BC_RL()	357
11.21.3.105 fsytrf_UP_RPM_BC_RL()	358

11.21.3.106 fsytrf_LOW_RPM_BC_Crout()	358
11.21.3.107 fsytrf_UP_RPM_BC_Crout()	358
11.21.3.108 fsytrf_UP_RPM()	358
11.21.3.109 fsytrf_nonunit() [2/3]	359
11.21.3.110 fsytrf_nonunit() [3/3]	359
11.21.3.111 fsytrf_RPM()	359
11.21.3.112 getTridiagonal()	359
11.21.3.113 LUdivine_gauss() [1/2]	360
11.21.3.114 LUdivine_small() [1/2]	360
11.21.3.115 LUdivine() [2/4]	360
11.21.3.116 LUdivine() [3/4]	361
11.21.3.117 MonotonicCompress()	361
11.21.3.118 MonotonicCompressMorePivots()	361
11.21.3.119 MonotonicCompressCycles()	361
11.21.3.120 MonotonicExpand()	362
11.21.3.121 applyP_block()	362
11.21.3.122 doApplyS()	362
11.21.3.123 MatrixApplyS() [1/3]	363
11.21.3.124 MatrixApplyS() [2/3]	363
11.21.3.125 MatrixApplyS() [3/3]	363
11.21.3.126 PermApplyS()	363
11.21.3.127 doApplyT()	364
11.21.3.128 MatrixApplyT() [1/3]	364
11.21.3.129 MatrixApplyT() [2/3]	364
11.21.3.130 MatrixApplyT() [3/3]	364
11.21.3.131 PermApplyT()	365
11.21.3.132 composePermutationsLLL()	365
11.21.3.133 composePermutationsLLM()	365
11.21.3.134 composePermutationsMLM()	366
11.21.3.135 cyclic_shift_mathPerm()	366
11.21.3.136 cyclic_shift_row_col() [1/2]	366
11.21.3.137 cyclic_shift_row() [1/3]	366
11.21.3.138 cyclic_shift_row() [2/3]	367
11.21.3.139 cyclic_shift_col() [1/3]	367
11.21.3.140 cyclic_shift_col() [2/3]	367
11.21.3.141 PLUQ_basecaseV3()	367
11.21.3.142 PLUQ_basecaseV2()	367
11.21.3.143 PLUQ_basecaseCrout()	368
11.21.3.144 _PLUQ()	368
11.21.3.145 PLUQ() [4/6]	368
11.21.3.146 threads_fgemm()	368
11.21.3.147 threads_ftrsm()	369

11.21.3.148 PLUQ() [5/6]	369
11.21.3.149 fflas_const_cast() [1/3]	369
11.21.3.150 fflas_const_cast() [2/3]	369
11.21.3.151 cyclic_shift_row_col() [2/2]	369
11.21.3.152 cyclic_shift_row() [3/3]	370
11.21.3.153 cyclic_shift_col() [3/3]	370
11.21.3.154 applyP() [4/4]	370
11.21.3.155 fgetrs() [3/4]	370
11.21.3.156 fgetrs() [4/4]	371
11.21.3.157 fgesv() [3/4]	371
11.21.3.158 fgesv() [4/4]	371
11.21.3.159 ftrtri() [2/2]	371
11.21.3.160 trinv_left() [2/2]	372
11.21.3.161 ftrtrm() [2/2]	372
11.21.3.162 PLUQ() [6/6]	372
11.21.3.163 LUdivine() [4/4]	372
11.21.3.164 LUdivine_small() [2/2]	373
11.21.3.165 LUdivine_gauss() [2/2]	373
11.21.3.166 RowEchelonForm() [3/3]	373
11.21.3.167 ReducedRowEchelonForm() [3/3]	373
11.21.3.168 ColumnEchelonForm() [3/3]	374
11.21.3.169 ReducedColumnEchelonForm() [3/3]	374
11.21.3.170 Invert() [3/4]	374
11.21.3.171 Invert() [4/4]	374
11.21.3.172 Invert2() [2/2]	374
11.21.3.173 CharPoly() [6/8]	375
11.21.3.174 CharPoly() [7/8]	375
11.21.3.175 CharPoly() [8/8]	375
11.21.3.176 MinPoly() [3/4]	375
11.21.3.177 MinPoly() [4/4]	375
11.21.3.178 MatVecMinPoly() [2/2]	376
11.21.3.179 KrylovElim()	376
11.21.3.180 SpecRankProfile()	376
11.21.3.181 Rank() [3/3]	376
11.21.3.182 IsSingular() [2/2]	376
11.21.3.183 Det() [5/6]	377
11.21.3.184 Det() [6/6]	377
11.21.3.185 Solve() [3/3]	377
11.21.3.186 solveLB() [2/2]	377
11.21.3.187 solveLB2() [2/2]	378
11.21.3.188 RandomNullSpaceVector() [3/3]	378
11.21.3.189 NullSpaceBasis() [2/2]	378

11.21.3.190 RowRankProfile()	[3/3]	378
11.21.3.191 ColumnRankProfile()	[3/3]	379
11.21.3.192 RowRankProfileSubmatrixIndices()	[2/2]	379
11.21.3.193 ColRankProfileSubmatrixIndices()	[2/2]	379
11.21.3.194 RowRankProfileSubmatrix()	[2/2]	379
11.21.3.195 ColRankProfileSubmatrix()	[2/2]	379
11.21.3.196 getTriangular< FFLAS_FIELD< FFLAS_ELT > >()	[1/2]	380
11.21.3.197 getTriangular< FFLAS_FIELD< FFLAS_ELT > >()	[2/2]	380
11.21.3.198 getEchelonForm< FFLAS_FIELD< FFLAS_ELT > >()	[1/2]	380
11.21.3.199 getEchelonForm< FFLAS_FIELD< FFLAS_ELT > >()	[2/2]	380
11.21.3.200 getEchelonTransform< FFLAS_FIELD< FFLAS_ELT > >()		381
11.21.3.201 getReducedEchelonForm< FFLAS_FIELD< FFLAS_ELT > >()	[1/2]	381
11.21.3.202 getReducedEchelonForm< FFLAS_FIELD< FFLAS_ELT > >()	[2/2]	381
11.21.3.203 getReducedEchelonTransform< FFLAS_FIELD< FFLAS_ELT > >()		381
11.21.3.204 LQUPtoInverseOfFullRankMinor()	[2/2]	382
11.21.3.205 fflas_const_cast()	[3/3]	382
11.21.3.206 failure()		382
11.21.3.207 isOdd()	[1/3]	382
11.21.3.208 isOdd()	[2/3]	382
11.21.3.209 isOdd()	[3/3]	382
11.21.3.210 NonZeroRandomMatrix()	[1/2]	383
11.21.3.211 NonZeroRandomMatrix()	[2/2]	383
11.21.3.212 RandomMatrix()	[1/2]	384
11.21.3.213 RandomMatrix()	[2/2]	384
11.21.3.214 RandomTriangularMatrix()	[1/2]	385
11.21.3.215 RandomTriangularMatrix()	[2/2]	385
11.21.3.216 RandInt()		386
11.21.3.217 RandomSymmetricMatrix()		386
11.21.3.218 RandomMatrixWithRank()	[1/2]	387
11.21.3.219 RandomMatrixWithRank()	[2/2]	387
11.21.3.220 RandomIndexSubset()		388
11.21.3.221 RandomPermutation()		388
11.21.3.222 RandomRankProfileMatrix()		388
11.21.3.223 swapval()		389
11.21.3.224 RandomSymmetricRankProfileMatrix()		389
11.21.3.225 RandomLTQSRankProfileMatrix()		389
11.21.3.226 RandomMatrixWithRankandRPM()	[1/2]	389
11.21.3.227 RandomMatrixWithRankandRPM()	[2/2]	390
11.21.3.228 RandomSymmetricMatrixWithRankandRPM()	[1/2]	390
11.21.3.229 RandomSymmetricMatrixWithRankandRPM()	[2/2]	391
11.21.3.230 RandomMatrixWithRankandRandomRPM()	[1/2]	392
11.21.3.231 RandomMatrixWithRankandRandomRPM()	[2/2]	392

11.21.3.232 RandomSymmetricMatrixWithRankandRandomRPM() [1/2]	393
11.21.3.233 RandomSymmetricMatrixWithRankandRandomRPM() [2/2]	393
11.21.3.234 RandomMatrixWithDet() [1/2]	394
11.21.3.235 RandomMatrixWithDet() [2/2]	394
11.21.3.236 RandomLTQSMMatrixWithRankandQSorder()	395
11.21.3.237 chooseField()	395
11.21.3.238 chooseField< Givaro::ZRing< int32_t > >()	395
11.21.3.239 chooseField< Givaro::ZRing< int64_t > >()	395
11.21.3.240 chooseField< Givaro::ZRing< float > >()	395
11.21.3.241 chooseField< Givaro::ZRing< double > >()	396
11.22 FFPACK::Protected Namespace Reference	396
11.22.1 Function Documentation	397
11.22.1.1 LUdivine_construct() [1/2]	397
11.22.1.2 GaussJordan()	398
11.22.1.3 KellerGehrig()	398
11.22.1.4 KGFast()	399
11.22.1.5 KGFast_generalized()	399
11.22.1.6 fgemv_kgf()	399
11.22.1.7 LUKrylov()	399
11.22.1.8 Danilevski()	399
11.22.1.9 RandomKrylovPrecond()	400
11.22.1.10 ArithProg()	400
11.22.1.11 LUKrylov_KGFast()	400
11.22.1.12 MatVecMinPoly()	400
11.22.1.13 Hybrid_KGF_LUK_MinPoly()	401
11.22.1.14 updateD()	401
11.22.1.15 newD()	401
11.22.1.16 CompressRows()	401
11.22.1.17 CompressRowsQK()	402
11.22.1.18 DeCompressRows()	402
11.22.1.19 DeCompressRowsQK()	402
11.22.1.20 CompressRowsQA()	402
11.22.1.21 DeCompressRowsQA()	403
11.22.1.22 LUdivine_construct() [2/2]	403
11.23 Givaro Namespace Reference	403
11.24 MKL_CONFIG Namespace Reference	403
11.25 Reclnt Namespace Reference	403
12 Data Structure Documentation	405
12.1 AlgoChooser< ModeT, ParSeq > Struct Template Reference	405
12.1.1 Member Typedef Documentation	405
12.1.1.1 value	405

12.2 AlgoChooser< ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeq > Struct Template Reference	405
12.2.1 Member Typedef Documentation	405
12.2.1.1 value	405
12.3 ALL< v > Struct Template Reference	405
12.4 ALL< false, v... > Struct Template Reference	406
12.4.1 Field Documentation	406
12.4.1.1 value	406
12.5 ALL< true, v... > Struct Template Reference	406
12.5.1 Field Documentation	406
12.5.1.1 value	406
12.6 ALL<> Struct Reference	406
12.6.1 Field Documentation	406
12.6.1.1 value	406
12.7 ArbitraryPrecIntTag Struct Reference	406
12.7.1 Detailed Description	406
12.8 AreEqual< X, Y > Class Template Reference	407
12.8.1 Field Documentation	407
12.8.1.1 value	407
12.9 AreEqual< X, X > Class Template Reference	407
12.9.1 Field Documentation	407
12.9.1.1 value	407
12.10 Argument Struct Reference	407
12.10.1 Field Documentation	407
12.10.1.1 c	407
12.10.1.2 example	407
12.10.1.3 helpString	408
12.10.1.4 type	408
12.10.1.5 data	408
12.11 associatedDelayedField< Field > Struct Template Reference	408
12.11.1 Member Typedef Documentation	408
12.11.1.1 field	408
12.11.1.2 type	408
12.12 associatedDelayedField< const FFPACK::RNSIntegerMod< RNS > > Struct Template Reference	408
12.12.1 Member Typedef Documentation	408
12.12.1.1 field	408
12.12.1.2 type	409
12.13 associatedDelayedField< const Givaro::Modular< T, X > > Struct Template Reference	409
12.13.1 Member Typedef Documentation	409
12.13.1.1 field	409
12.13.1.2 type	409
12.14 associatedDelayedField< const Givaro::ModularBalanced< T > > Struct Template Reference	409

12.14.1 Member Typedef Documentation	409
12.14.1.1 field	409
12.14.1.2 type	409
12.15 associatedDelayedField< const Givaro::ZRing< T > > Struct Template Reference	410
12.15.1 Member Typedef Documentation	410
12.15.1.1 field	410
12.15.1.2 type	410
12.16 Auto Struct Reference	410
12.17 Bench< Elt > Class Template Reference	410
12.17.1 Member Typedef Documentation	411
12.17.1.1 Field	411
12.17.1.2 Elt_ptr	411
12.17.1.3 Residu	411
12.17.1.4 enable_if_t	411
12.17.1.5 is_same_element	411
12.17.1.6 enable_if_no_simd_t	411
12.17.1.7 enable_if_simd128_t	411
12.17.1.8 enable_if_simd256_t	411
12.17.1.9 enable_if_simd512_t	412
12.17.2 Constructor & Destructor Documentation	412
12.17.2.1 Bench()	412
12.17.3 Member Function Documentation	412
12.17.3.1 cardinality() [1/2]	412
12.17.3.2 cardinality() [2/2]	412
12.17.3.3 doBenches()	412
12.17.3.4 run()	412
12.17.4 Field Documentation	412
12.17.4.1 F	412
12.17.4.2 m	412
12.17.4.3 n	413
12.17.4.4 iters	413
12.17.4.5 inplace	413
12.18 Bini Struct Reference	413
12.19 Block Struct Reference	413
12.20 BlockTransposeSIMD< Field, Simd, > Struct Template Reference	413
12.20.1 Member Function Documentation	413
12.20.1.1 size()	413
12.20.1.2 info()	414
12.20.1.3 transpose() [1/5]	414
12.20.1.4 transpose() [2/5]	414
12.20.1.5 transpose() [3/5]	414
12.20.1.6 transpose() [4/5]	414

12.20.1.7 transpose() [5/5]	414
12.21 callLUdivine_small< Element > Class Template Reference	415
12.21.1 Member Function Documentation	415
12.21.1.1 operator()()	415
12.22 callLUdivine_small< double > Class Reference	415
12.22.1 Member Function Documentation	415
12.22.1.1 operator()()	415
12.23 callLUdivine_small< float > Class Reference	416
12.23.1 Member Function Documentation	416
12.23.1.1 operator()()	416
12.24 CharpolyFailed Class Reference	416
12.25 Checker_Empty< Field > Struct Template Reference	416
12.25.1 Constructor & Destructor Documentation	417
12.25.1.1 Checker_Empty()	417
12.25.2 Member Function Documentation	417
12.25.2.1 check()	417
12.26 CheckerImplem_charpoly< Field, Polynomial > Class Template Reference	417
12.26.1 Constructor & Destructor Documentation	417
12.26.1.1 CheckerImplem_charpoly() [1/2]	417
12.26.1.2 CheckerImplem_charpoly() [2/2]	417
12.26.1.3 ~CheckerImplem_charpoly()	417
12.26.2 Member Function Documentation	418
12.26.2.1 check()	418
12.27 CheckerImplem_charpoly< Givaro::ZRing< Givaro::Integer >, Polynomial > Class Template Reference	418
12.27.1 Member Typedef Documentation	418
12.27.1.1 Ring	418
12.27.2 Constructor & Destructor Documentation	418
12.27.2.1 CheckerImplem_charpoly() [1/2]	418
12.27.2.2 CheckerImplem_charpoly() [2/2]	418
12.27.2.3 ~CheckerImplem_charpoly()	418
12.27.3 Member Function Documentation	419
12.27.3.1 check()	419
12.28 CheckerImplem_Det< Field > Class Template Reference	419
12.28.1 Constructor & Destructor Documentation	419
12.28.1.1 CheckerImplem_Det() [1/2]	419
12.28.1.2 CheckerImplem_Det() [2/2]	419
12.28.1.3 ~CheckerImplem_Det()	419
12.28.2 Member Function Documentation	419
12.28.2.1 check()	419
12.29 CheckerImplem_fgemm< Field > Class Template Reference	421
12.29.1 Constructor & Destructor Documentation	421

12.29.1.1 CheckerImplem_fgemm() [1/2]	421
12.29.1.2 CheckerImplem_fgemm() [2/2]	421
12.29.1.3 ~CheckerImplem_fgemm()	421
12.29.2 Member Function Documentation	422
12.29.2.1 check()	422
12.30 CheckerImplem_ftsm< Field > Class Template Reference	422
12.30.1 Constructor & Destructor Documentation	422
12.30.1.1 CheckerImplem_ftsm() [1/2]	422
12.30.1.2 CheckerImplem_ftsm() [2/2]	422
12.30.1.3 ~CheckerImplem_ftsm()	422
12.30.2 Member Function Documentation	423
12.30.2.1 check()	423
12.31 CheckerImplem_invert< Field > Class Template Reference	423
12.31.1 Constructor & Destructor Documentation	423
12.31.1.1 CheckerImplem_invert() [1/2]	423
12.31.1.2 CheckerImplem_invert() [2/2]	423
12.31.1.3 ~CheckerImplem_invert()	423
12.31.2 Member Function Documentation	423
12.31.2.1 check()	423
12.32 CheckerImplem_PLUQ< Field > Class Template Reference	424
12.32.1 Constructor & Destructor Documentation	424
12.32.1.1 CheckerImplem_PLUQ() [1/2]	424
12.32.1.2 CheckerImplem_PLUQ() [2/2]	424
12.32.1.3 ~CheckerImplem_PLUQ()	424
12.32.2 Member Function Documentation	424
12.32.2.1 check()	424
12.33 Classic Struct Reference	425
12.34 Column Struct Reference	425
12.35 CompactElement< Element > Struct Template Reference	425
12.35.1 Member Typedef Documentation	425
12.35.1.1 type	425
12.36 CompactElement< double > Struct Reference	425
12.36.1 Member Typedef Documentation	425
12.36.1.1 type	425
12.37 CompactElement< float > Struct Reference	426
12.37.1 Member Typedef Documentation	426
12.37.1.1 type	426
12.38 CompactElement< int16_t > Struct Reference	426
12.38.1 Member Typedef Documentation	426
12.38.1.1 type	426
12.39 CompactElement< int32_t > Struct Reference	426
12.39.1 Member Typedef Documentation	426

12.39.1.1 type	426
12.40 CompactElement< int64_t > Struct Reference	426
12.40.1 Member Typedef Documentation	426
12.40.1.1 type	426
12.41 compatible_data_type< Field > Struct Template Reference	427
12.41.1 Field Documentation	427
12.41.1.1 value	427
12.42 compatible_data_type< Givaro::ZRing< double > > Struct Reference	427
12.42.1 Field Documentation	427
12.42.1.1 value	427
12.43 compatible_data_type< Givaro::ZRing< float > > Struct Reference	427
12.43.1 Field Documentation	427
12.43.1.1 value	427
12.44 Compose< H1, H2 > Struct Template Reference	427
12.44.1 Constructor & Destructor Documentation	428
12.44.1.1 Compose() [1/5]	428
12.44.1.2 Compose() [2/5]	428
12.44.1.3 Compose() [3/5]	428
12.44.1.4 Compose() [4/5]	428
12.44.1.5 Compose() [5/5]	428
12.44.2 Member Function Documentation	428
12.44.2.1 first_component()	428
12.44.2.2 second_component()	428
12.44.3 Friends And Related Symbol Documentation	428
12.44.3.1 operator<<	428
12.45 Simd128_impl< true, true, false, 2 >::Converter Union Reference	429
12.45.1 Field Documentation	429
12.45.1.1 v	429
12.45.1.2 t	429
12.46 Simd128_impl< true, true, false, 4 >::Converter Union Reference	429
12.46.1 Field Documentation	429
12.46.1.1 v	429
12.46.1.2 t	429
12.47 Simd128_impl< true, true, false, 8 >::Converter Union Reference	429
12.47.1 Field Documentation	429
12.47.1.1 v	429
12.47.1.2 t	429
12.48 Simd128_impl< true, true, true, 2 >::Converter Union Reference	430
12.48.1 Field Documentation	430
12.48.1.1 v	430
12.48.1.2 t	430
12.49 Simd128_impl< true, true, true, 4 >::Converter Union Reference	430

12.49.1 Field Documentation	430
12.49.1.1 v	430
12.49.1.2 t	430
12.50 Simd128_impl< true, true, true, 8 >::Converter Union Reference	430
12.50.1 Field Documentation	430
12.50.1.1 v	430
12.50.1.2 t	430
12.51 Simd256_impl< true, false, true, 8 >::Converter Union Reference	431
12.51.1 Field Documentation	431
12.51.1.1 v	431
12.51.1.2 t	431
12.52 Simd256_impl< true, true, false, 2 >::Converter Union Reference	431
12.52.1 Field Documentation	431
12.52.1.1 v	431
12.52.1.2 t	431
12.53 Simd256_impl< true, true, false, 4 >::Converter Union Reference	431
12.53.1 Field Documentation	431
12.53.1.1 v	431
12.53.1.2 t	431
12.54 Simd256_impl< true, true, false, 8 >::Converter Union Reference	432
12.54.1 Field Documentation	432
12.54.1.1 v	432
12.54.1.2 t	432
12.55 Simd256_impl< true, true, true, 2 >::Converter Union Reference	432
12.55.1 Field Documentation	432
12.55.1.1 v	432
12.55.1.2 t	432
12.56 Simd256_impl< true, true, true, 4 >::Converter Union Reference	432
12.56.1 Field Documentation	432
12.56.1.1 v	432
12.56.1.2 t	432
12.57 Simd256_impl< true, true, true, 8 >::Converter Union Reference	433
12.57.1 Field Documentation	433
12.57.1.1 v	433
12.57.1.2 t	433
12.58 Simd512_impl< true, true, false, 8 >::Converter Union Reference	433
12.58.1 Field Documentation	433
12.58.1.1 v	433
12.58.1.2 t	433
12.59 Simd512_impl< true, true, true, 8 >::Converter Union Reference	433
12.59.1 Field Documentation	433
12.59.1.1 v	433

12.59.1.2 t	433
12.60 ConvertTo< T > Struct Template Reference	434
12.60.1 Detailed Description	434
12.61 Coo< ValT, IdxT > Struct Template Reference	434
12.61.1 Member Typedef Documentation	434
12.61.1.1 Self	434
12.61.2 Constructor & Destructor Documentation	434
12.61.2.1 Coo() [1/4]	434
12.61.2.2 Coo() [2/4]	435
12.61.2.3 Coo() [3/4]	435
12.61.2.4 Coo() [4/4]	435
12.61.3 Member Function Documentation	435
12.61.3.1 operator=() [1/2]	435
12.61.3.2 operator=() [2/2]	435
12.61.4 Field Documentation	435
12.61.4.1 val	435
12.61.4.2 row	435
12.61.4.3 col	435
12.62 Coo< Field > Struct Template Reference	435
12.62.1 Constructor & Destructor Documentation	436
12.62.1.1 Coo() [1/4]	436
12.62.1.2 Coo() [2/4]	436
12.62.1.3 Coo() [3/4]	436
12.62.1.4 Coo() [4/4]	436
12.62.2 Member Function Documentation	436
12.62.2.1 operator=() [1/2]	436
12.62.2.2 operator=() [2/2]	436
12.62.3 Field Documentation	436
12.62.3.1 val	436
12.62.3.2 col	437
12.62.3.3 row	437
12.62.3.4 deleted	437
12.63 Coo< ValT, IdxT > Struct Template Reference	437
12.63.1 Member Typedef Documentation	437
12.63.1.1 Self	437
12.63.2 Constructor & Destructor Documentation	437
12.63.2.1 Coo() [1/4]	437
12.63.2.2 Coo() [2/4]	437
12.63.2.3 Coo() [3/4]	438
12.63.2.4 Coo() [4/4]	438
12.63.3 Member Function Documentation	438
12.63.3.1 operator=() [1/2]	438

12.63.3.2 operator=() [2/2]	438
12.63.4 Field Documentation	438
12.63.4.1 val	438
12.63.4.2 row	438
12.63.4.3 col	438
12.64 CooMat< Field > Struct Template Reference	438
12.64.1 Field Documentation	439
12.64.1.1 _coo16	439
12.64.1.2 _coo32	439
12.64.1.3 _coo64	439
12.64.1.4 _coo16_zo	439
12.64.1.5 _coo32_zo	439
12.64.1.6 _coo64_zo	439
12.65 count_nonconst_lvalue_reference< T > Struct Template Reference	439
12.66 count_nonconst_lvalue_reference< const T &, O... > Struct Template Reference	439
12.66.1 Field Documentation	439
12.66.1.1 n	439
12.67 count_nonconst_lvalue_reference< T &, O... > Struct Template Reference	440
12.67.1 Field Documentation	440
12.67.1.1 n	440
12.68 count_nonconst_lvalue_reference< T, O... > Struct Template Reference	440
12.68.1 Field Documentation	440
12.68.1.1 n	440
12.69 count_nonconst_lvalue_reference<> Struct Reference	440
12.69.1 Field Documentation	440
12.69.1.1 n	440
12.70 CsrMat< Field > Struct Template Reference	440
12.70.1 Field Documentation	441
12.70.1.1 _csr16	441
12.70.1.2 _csr32	441
12.70.1.3 _csr64	441
12.70.1.4 _csr16_zo	441
12.70.1.5 _csr32_zo	441
12.70.1.6 _csr64_zo	441
12.71 DefaultBoundedTag Struct Reference	441
12.71.1 Detailed Description	441
12.72 DefaultTag Struct Reference	441
12.72.1 Detailed Description	442
12.73 DelayedTag Struct Reference	442
12.73.1 Detailed Description	442
12.74 DivideAndConquer Struct Reference	442
12.75 ElementTraits< Element > Struct Template Reference	442

12.75.1 Detailed Description	442
12.75.2 Member Typedef Documentation	442
12.75.2.1 value	442
12.76 ElementTraits< double > Struct Reference	442
12.76.1 Member Typedef Documentation	443
12.76.1.1 value	443
12.77 ElementTraits< FFPACK::rns_double_elt > Struct Reference	443
12.77.1 Member Typedef Documentation	443
12.77.1.1 value	443
12.78 ElementTraits< float > Struct Reference	443
12.78.1 Member Typedef Documentation	443
12.78.1.1 value	443
12.79 ElementTraits< Givaro::Integer > Struct Reference	443
12.79.1 Member Typedef Documentation	443
12.79.1.1 value	443
12.80 ElementTraits< int16_t > Struct Reference	444
12.80.1 Member Typedef Documentation	444
12.80.1.1 value	444
12.81 ElementTraits< int32_t > Struct Reference	444
12.81.1 Member Typedef Documentation	444
12.81.1.1 value	444
12.82 ElementTraits< int64_t > Struct Reference	444
12.82.1 Member Typedef Documentation	444
12.82.1.1 value	444
12.83 ElementTraits< int8_t > Struct Reference	444
12.83.1 Member Typedef Documentation	445
12.83.1.1 value	445
12.84 ElementTraits< RecInt::rint< K > > Struct Template Reference	445
12.84.1 Member Typedef Documentation	445
12.84.1.1 value	445
12.85 ElementTraits< RecInt::rmint< K, MG > > Struct Template Reference	445
12.85.1 Member Typedef Documentation	445
12.85.1.1 value	445
12.86 ElementTraits< RecInt::ruint< K > > Struct Template Reference	445
12.86.1 Member Typedef Documentation	446
12.86.1.1 value	446
12.87 ElementTraits< uint16_t > Struct Reference	446
12.87.1 Member Typedef Documentation	446
12.87.1.1 value	446
12.88 ElementTraits< uint32_t > Struct Reference	446
12.88.1 Member Typedef Documentation	446
12.88.1.1 value	446

12.89 ElementTraits< uint64_t > Struct Reference	446
12.89.1 Member Typedef Documentation	446
12.89.1.1 value	446
12.90 ElementTraits< uint8_t > Struct Reference	447
12.90.1 Member Typedef Documentation	447
12.90.1.1 value	447
12.91 EllMat< Field > Struct Template Reference	447
12.91.1 Field Documentation	447
12.91.1.1 _ell16	447
12.91.1.2 _ell32	447
12.91.1.3 _ell64	447
12.91.1.4 _ell16_zo	447
12.91.1.5 _ell32_zo	447
12.91.1.6 _ell64_zo	448
12.92 Failure Class Reference	448
12.92.1 Detailed Description	448
12.92.2 Constructor & Destructor Documentation	448
12.92.2.1 Failure()	448
12.92.3 Member Function Documentation	448
12.92.3.1 operator>() [1/2]	448
12.92.3.2 operator>() [2/2]	448
12.92.3.3 setErrorStream()	449
12.92.3.4 print()	449
12.92.4 Field Documentation	449
12.92.4.1 _errorStream	449
12.93 FailureCharpolyCheck Class Reference	449
12.94 FailureDetCheck Class Reference	449
12.95 FailureFgemmCheck Class Reference	449
12.96 FailureInvertCheck Class Reference	450
12.97 FailurePLUQCheck Class Reference	450
12.98 FailureTrsmCheck Class Reference	450
12.99 FieldSimd< _Field > Class Template Reference	450
12.99.1 Member Typedef Documentation	451
12.99.1.1 Field	451
12.99.1.2 Element	451
12.99.1.3 simd	451
12.99.1.4 vect_t	451
12.99.1.5 scalar_t	451
12.99.2 Constructor & Destructor Documentation	451
12.99.2.1 FieldSimd() [1/3]	451
12.99.2.2 FieldSimd() [2/3]	452
12.99.2.3 FieldSimd() [3/3]	452

12.99.3 Member Function Documentation	452
12.99.3.1 operator=() [1/2]	452
12.99.3.2 operator=() [2/2]	452
12.99.3.3 init() [1/2]	452
12.99.3.4 init() [2/2]	452
12.99.3.5 add() [1/2]	452
12.99.3.6 add() [2/2]	452
12.99.3.7 addin()	452
12.99.3.8 add_r() [1/2]	453
12.99.3.9 add_r() [2/2]	453
12.99.3.10 addin_r()	453
12.99.3.11 sub() [1/2]	453
12.99.3.12 sub() [2/2]	453
12.99.3.13 subin()	453
12.99.3.14 sub_r() [1/2]	453
12.99.3.15 sub_r() [2/2]	453
12.99.3.16 subin_r()	454
12.99.3.17 zero() [1/2]	454
12.99.3.18 zero() [2/2]	454
12.99.3.19 mod()	454
12.99.3.20 mul() [1/2]	454
12.99.3.21 mul() [2/2]	454
12.99.3.22 mulin()	454
12.99.3.23 mul_r() [1/2]	454
12.99.3.24 mul_r() [2/2]	454
12.99.3.25 axpy() [1/2]	455
12.99.3.26 axpy() [2/2]	455
12.99.3.27 axpyin()	455
12.99.3.28 axpy_r() [1/2]	455
12.99.3.29 axpy_r() [2/2]	455
12.99.3.30 axpyin_r()	455
12.99.3.31 maxpy() [1/2]	455
12.99.3.32 maxpy() [2/2]	456
12.99.3.33 maxpyin()	456
12.99.4 Field Documentation	456
12.99.4.1 vect_size	456
12.99.4.2 alignment	456
12.100 FieldTraits< Field > Struct Template Reference	456
12.100.1 Detailed Description	456
12.100.2 Member Typedef Documentation	456
12.100.2.1 category	456
12.100.3 Field Documentation	457

12.100.3.1 balanced	457
12.101 FieldTraits< FFPACK::RNSInteger< T > > Struct Template Reference	457
12.101.1 Member Typedef Documentation	457
12.101.1.1 category	457
12.101.2 Field Documentation	457
12.101.2.1 balanced	457
12.102 FieldTraits< FFPACK::RNSIntegerMod< T > > Struct Template Reference	457
12.102.1 Member Typedef Documentation	457
12.102.1.1 category	457
12.102.2 Field Documentation	458
12.102.2.1 balanced	458
12.103 FieldTraits< Givaro::Modular< Element > > Struct Template Reference	458
12.103.1 Member Typedef Documentation	458
12.103.1.1 category	458
12.103.2 Field Documentation	458
12.103.2.1 balanced	458
12.104 FieldTraits< Givaro::ModularBalanced< Element > > Struct Template Reference	458
12.104.1 Member Typedef Documentation	458
12.104.1.1 category	458
12.104.2 Field Documentation	459
12.104.2.1 balanced	459
12.105 FieldTraits< Givaro::ZRing< double > > Struct Reference	459
12.105.1 Member Typedef Documentation	459
12.105.1.1 category	459
12.105.2 Field Documentation	459
12.105.2.1 balanced	459
12.106 FieldTraits< Givaro::ZRing< float > > Struct Reference	459
12.106.1 Member Typedef Documentation	459
12.106.1.1 category	459
12.106.2 Field Documentation	460
12.106.2.1 balanced	460
12.107 FieldTraits< Givaro::ZRing< Givaro::Integer > > Struct Reference	460
12.107.1 Member Typedef Documentation	460
12.107.1.1 category	460
12.107.2 Field Documentation	460
12.107.2.1 balanced	460
12.108 FieldTraits< Givaro::ZRing< int16_t > > Struct Reference	460
12.108.1 Member Typedef Documentation	460
12.108.1.1 category	460
12.108.2 Field Documentation	461
12.108.2.1 balanced	461
12.109 FieldTraits< Givaro::ZRing< int32_t > > Struct Reference	461

12.109.1 Member Typedef Documentation	461
12.109.1.1 category	461
12.109.2 Field Documentation	461
12.109.2.1 balanced	461
12.110 FieldTraits< Givaro::ZRing< int64_t > > Struct Reference	461
12.110.1 Member Typedef Documentation	461
12.110.1.1 category	461
12.110.2 Field Documentation	461
12.110.2.1 balanced	461
12.111 FieldTraits< Givaro::ZRing< RecInt::ruint< K > > > Struct Template Reference	462
12.111.1 Member Typedef Documentation	462
12.111.1.1 category	462
12.111.2 Field Documentation	462
12.111.2.1 balanced	462
12.112 FieldTraits< Givaro::ZRing< uint16_t > > Struct Reference	462
12.112.1 Member Typedef Documentation	462
12.112.1.1 category	462
12.112.2 Field Documentation	462
12.112.2.1 balanced	462
12.113 FieldTraits< Givaro::ZRing< uint32_t > > Struct Reference	462
12.113.1 Member Typedef Documentation	463
12.113.1.1 category	463
12.113.2 Field Documentation	463
12.113.2.1 balanced	463
12.114 FieldTraits< Givaro::ZRing< uint64_t > > Struct Reference	463
12.114.1 Member Typedef Documentation	463
12.114.1.1 category	463
12.114.2 Field Documentation	463
12.114.2.1 balanced	463
12.115 Fixed Struct Reference	463
12.116 FixedPrecIntTag Struct Reference	463
12.116.1 Detailed Description	464
12.117 ScalFunctionsBase< Element, typename enable_if< is_floating_point< Element >::value >::type >::FloatingPointTestDistribution Class Reference	464
12.117.1 Member Typedef Documentation	464
12.117.1.1 IntType	464
12.117.2 Constructor & Destructor Documentation	464
12.117.2.1 FloatingPointTestDistribution()	464
12.117.3 Member Function Documentation	464
12.117.3.1 operator>()	464
12.118 ftrmmLeftLowerNoTransNonUnit< Element > Class Template Reference	464
12.119 ftrmmLeftLowerNoTransUnit< Element > Class Template Reference	464

12.120 ftrmmLeftLowerTransNonUnit< Element > Class Template Reference	465
12.121 ftrmmLeftLowerTransUnit< Element > Class Template Reference	465
12.122 ftrmmLeftUpperNoTransNonUnit< Element > Class Template Reference	465
12.123 ftrmmLeftUpperNoTransUnit< Element > Class Template Reference	465
12.124 ftrmmLeftUpperTransNonUnit< Element > Class Template Reference	465
12.125 ftrmmLeftUpperTransUnit< Element > Class Template Reference	465
12.126 ftrmmRightLowerNoTransNonUnit< Element > Class Template Reference	465
12.127 ftrmmRightLowerNoTransUnit< Element > Class Template Reference	465
12.128 ftrmmRightLowerTransNonUnit< Element > Class Template Reference	466
12.129 ftrmmRightLowerTransUnit< Element > Class Template Reference	466
12.130 ftrmmRightUpperNoTransNonUnit< Element > Class Template Reference	466
12.131 ftrmmRightUpperNoTransUnit< Element > Class Template Reference	466
12.132 ftrmmRightUpperTransNonUnit< Element > Class Template Reference	466
12.133 ftrmmRightUpperTransUnit< Element > Class Template Reference	466
12.134 ftrsmLeftLowerNoTransNonUnit< Element > Class Template Reference	466
12.135 ftrsmLeftLowerNoTransUnit< Element > Class Template Reference	466
12.136 ftrsmLeftLowerTransNonUnit< Element > Class Template Reference	467
12.137 ftrsmLeftLowerTransUnit< Element > Class Template Reference	467
12.138 ftrsmLeftUpperNoTransNonUnit< Element > Class Template Reference	467
12.138.1 Detailed Description	467
12.139 ftrsmLeftUpperNoTransUnit< Element > Class Template Reference	467
12.140 ftrsmLeftUpperTransNonUnit< Element > Class Template Reference	467
12.141 ftrsmLeftUpperTransUnit< Element > Class Template Reference	467
12.142 ftrsmRightLowerNoTransNonUnit< Element > Class Template Reference	468
12.143 ftrsmRightLowerNoTransUnit< Element > Class Template Reference	468
12.144 ftrsmRightLowerTransNonUnit< Element > Class Template Reference	468
12.145 ftrsmRightLowerTransUnit< Element > Class Template Reference	468
12.146 ftrsmRightUpperNoTransNonUnit< Element > Class Template Reference	468
12.147 ftrsmRightUpperNoTransUnit< Element > Class Template Reference	468
12.148 ftrsmRightUpperTransNonUnit< Element > Class Template Reference	468
12.149 ftrsmRightUpperTransUnit< Element > Class Template Reference	468
12.150 GenericTag Struct Reference	469
12.150.1 Detailed Description	469
12.151 GenericTag Struct Reference	469
12.151.1 Detailed Description	469
12.152 Grain Struct Reference	469
12.153 has_minus_eq_impl< C > Struct Template Reference	469
12.153.1 Field Documentation	469
12.153.1.1 value	469
12.154 has_minus_impl< C > Struct Template Reference	469
12.154.1 Field Documentation	470
12.154.1.1 value	470

12.155 has_mul_eq_impl< C > Struct Template Reference	470
12.155.1 Field Documentation	470
12.155.1.1 value	470
12.156 has_mul_impl< C > Struct Template Reference	470
12.156.1 Field Documentation	470
12.156.1.1 value	470
12.157 has_operation< T > Struct Template Reference	470
12.157.1 Field Documentation	471
12.157.1.1 value	471
12.158 has_plus_eq_impl< C > Struct Template Reference	471
12.158.1 Field Documentation	471
12.158.1.1 value	471
12.159 has_plus_impl< C > Struct Template Reference	471
12.159.1 Field Documentation	471
12.159.1.1 value	471
12.160 HelperFlag Struct Reference	471
12.160.1 Field Documentation	472
12.160.1.1 none	472
12.160.1.2 coo	472
12.160.1.3 csr	472
12.160.1.4 ell	472
12.160.1.5 aut	472
12.160.1.6 pm1	472
12.161 HelperMod< Field, ElementTraits > Struct Template Reference	472
12.162 HelperMod< Field, ElementCategories::MachineIntTag > Struct Template Reference	472
12.162.1 Constructor & Destructor Documentation	472
12.162.1.1 HelperMod() [1/2]	472
12.162.1.2 HelperMod() [2/2]	473
12.162.2 Field Documentation	473
12.162.2.1 p	473
12.162.2.2 invp	473
12.162.2.3 min	473
12.162.2.4 max	473
12.162.2.5 pow50rem	473
12.163 HelperMod< Field, FFLAS::ElementCategories::ArbitraryPrecIntTag > Struct Template Reference	473
12.163.1 Constructor & Destructor Documentation	473
12.163.1.1 HelperMod() [1/2]	473
12.163.1.2 HelperMod() [2/2]	474
12.163.2 Field Documentation	474
12.163.2.1 p	474
12.164 HelperMod< Field, FFLAS::ElementCategories::FixedPrecIntTag > Struct Template Reference	474
12.164.1 Constructor & Destructor Documentation	474

12.164.1.1 HelperMod() [1/2]	474
12.164.1.2 HelperMod() [2/2]	474
12.164.2 Field Documentation	474
12.164.2.1 p	474
12.165 HelperMod< Field, FFLAS::ElementCategories::MachineFloatTag > Struct Template Reference	474
12.165.1 Constructor & Destructor Documentation	475
12.165.1.1 HelperMod() [1/2]	475
12.165.1.2 HelperMod() [2/2]	475
12.165.2 Field Documentation	475
12.165.2.1 p	475
12.165.2.2 invp	475
12.165.2.3 min	475
12.165.2.4 max	475
12.166 Hybrid Struct Reference	475
12.167 Info Struct Reference	475
12.167.1 Constructor & Destructor Documentation	476
12.167.1.1 Info() [1/4]	476
12.167.1.2 Info() [2/4]	476
12.167.1.3 Info() [3/4]	476
12.167.1.4 Info() [4/4]	476
12.167.2 Member Function Documentation	476
12.167.2.1 operator=() [1/2]	476
12.167.2.2 operator=() [2/2]	476
12.167.3 Field Documentation	476
12.167.3.1 size	476
12.167.3.2 perm	476
12.167.3.3 begin	476
12.168 Info Struct Reference	477
12.168.1 Constructor & Destructor Documentation	477
12.168.1.1 Info() [1/4]	477
12.168.1.2 Info() [2/4]	477
12.168.1.3 Info() [3/4]	477
12.168.1.4 Info() [4/4]	477
12.168.2 Member Function Documentation	477
12.168.2.1 operator=() [1/2]	477
12.168.2.2 operator=() [2/2]	477
12.168.3 Field Documentation	477
12.168.3.1 size	477
12.168.3.2 perm	477
12.168.3.3 begin	478
12.169 is_all_same< Args > Struct Template Reference	478
12.170 is_all_same< T, Args... > Struct Template Reference	478

12.170.1 Field Documentation	478
12.170.1.1 value	478
12.171 is_all_same<> Struct Reference	478
12.171.1 Field Documentation	478
12.171.1.1 value	478
12.172 is_simd< T > Struct Template Reference	478
12.172.1 Member Typedef Documentation	479
12.172.1.1 type	479
12.172.2 Field Documentation	479
12.172.2.1 value	479
12.173 isSparseMatrix< Field, M > Struct Template Reference	479
12.174 isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::COO > > Struct Template Reference	479
12.175 isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::COO_ZO > > Struct Template Reference	479
12.176 isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR > > Struct Template Reference	480
12.177 isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR_HYB > > Struct Template Reference	480
12.178 isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR_ZO > > Struct Template Reference	480
12.179 isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL > > Struct Template Reference	481
12.180 isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_simd > > Struct Template Reference	481
12.181 isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_simd_ZO > > Struct Template Reference	481
12.182 isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_ZO > > Struct Template Reference	482
12.183 isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::HYB_ZO > > Struct Template Reference	482
12.184 isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::SELL > > Struct Template Reference	482
12.185 isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::SELL_ZO > > Struct Template Reference	482
12.186 isSparseMatrixMKLFormat< F, M > Struct Template Reference	483
12.187 isSparseMatrixSimdFormat< F, M > Struct Template Reference	483
12.188 isZOSparseMatrix< F, M > Struct Template Reference	483
12.189 isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::COO_ZO > > Struct Template Reference	484
12.190 isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR_ZO > > Struct Template Reference	484
12.191 isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_simd_ZO > > Struct Template Reference	484
12.192 isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_ZO > > Struct Template Reference	484
12.193 isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::SELL_ZO > > Struct Template Reference	485
12.194 Iterative Struct Reference	485
12.195 LazyTag Struct Reference	485
12.195.1 Detailed Description	485
12.196 limits< T > Struct Template Reference	485
12.197 limits< char > Struct Reference	485
12.197.1 Member Typedef Documentation	486
12.197.1.1 T	486
12.197.2 Member Function Documentation	486

12.197.2.1 max()	486
12.197.2.2 min()	486
12.197.2.3 digits()	486
12.198 limits< double > Struct Reference	486
12.198.1 Member Typedef Documentation	486
12.198.1.1 T	486
12.198.2 Member Function Documentation	486
12.198.2.1 max()	486
12.198.2.2 min()	487
12.198.2.3 digits()	487
12.199 limits< float > Struct Reference	487
12.199.1 Member Typedef Documentation	487
12.199.1.1 T	487
12.199.2 Member Function Documentation	487
12.199.2.1 max()	487
12.199.2.2 min()	487
12.199.2.3 digits()	487
12.200 limits< Givaro::Integer > Struct Reference	487
12.200.1 Member Typedef Documentation	488
12.200.1.1 T	488
12.200.2 Member Function Documentation	488
12.200.2.1 max()	488
12.200.2.2 min()	488
12.201 limits< int > Struct Reference	488
12.201.1 Member Typedef Documentation	488
12.201.1.1 T	488
12.201.2 Member Function Documentation	488
12.201.2.1 max()	488
12.201.2.2 min()	488
12.201.2.3 digits()	488
12.202 limits< long > Struct Reference	488
12.202.1 Member Typedef Documentation	489
12.202.1.1 T	489
12.202.2 Member Function Documentation	489
12.202.2.1 max()	489
12.202.2.2 min()	489
12.202.2.3 digits()	489
12.203 limits< long long > Struct Reference	489
12.203.1 Member Typedef Documentation	489
12.203.1.1 T	489
12.203.2 Member Function Documentation	489
12.203.2.1 max()	489

12.203.2.2 min()	490
12.203.2.3 digits()	490
12.204 limits< Reclnt::rint< K > > Struct Template Reference	490
12.204.1 Member Typedef Documentation	490
12.204.1.1 T	490
12.204.2 Member Function Documentation	490
12.204.2.1 max()	490
12.204.2.2 min()	490
12.205 limits< Reclnt::ruint< K > > Struct Template Reference	490
12.205.1 Member Typedef Documentation	491
12.205.1.1 T	491
12.205.2 Member Function Documentation	491
12.205.2.1 max()	491
12.205.2.2 min()	491
12.206 limits< short int > Struct Reference	491
12.206.1 Member Typedef Documentation	491
12.206.1.1 T	491
12.206.2 Member Function Documentation	491
12.206.2.1 max()	491
12.206.2.2 min()	491
12.206.2.3 digits()	491
12.207 limits< signed char > Struct Reference	492
12.207.1 Member Typedef Documentation	492
12.207.1.1 T	492
12.207.2 Member Function Documentation	492
12.207.2.1 max()	492
12.207.2.2 min()	492
12.207.2.3 digits()	492
12.208 limits< unsigned char > Struct Reference	492
12.208.1 Member Typedef Documentation	492
12.208.1.1 T	492
12.208.2 Member Function Documentation	493
12.208.2.1 max()	493
12.208.2.2 min()	493
12.208.2.3 digits()	493
12.209 limits< unsigned int > Struct Reference	493
12.209.1 Member Typedef Documentation	493
12.209.1.1 T	493
12.209.2 Member Function Documentation	493
12.209.2.1 max()	493
12.209.2.2 min()	493
12.209.2.3 digits()	493

12.210 limits< unsigned long > Struct Reference	493
12.210.1 Member Typedef Documentation	494
12.210.1.1 T	494
12.210.2 Member Function Documentation	494
12.210.2.1 max()	494
12.210.2.2 min()	494
12.210.2.3 digits()	494
12.211 limits< unsigned long long > Struct Reference	494
12.211.1 Member Typedef Documentation	494
12.211.1.1 T	494
12.211.2 Member Function Documentation	494
12.211.2.1 max()	494
12.211.2.2 min()	494
12.211.2.3 digits()	495
12.212 limits< unsigned short int > Struct Reference	495
12.212.1 Member Typedef Documentation	495
12.212.1.1 T	495
12.212.2 Member Function Documentation	495
12.212.2.1 max()	495
12.212.2.2 min()	495
12.212.2.3 digits()	495
12.213 MachineFloatTag Struct Reference	495
12.213.1 Detailed Description	495
12.214 MachineIntTag Struct Reference	495
12.214.1 Detailed Description	496
12.215 MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait > Struct Template Reference	496
12.215.1 Member Typedef Documentation	497
12.215.1.1 Self_t	497
12.215.1.2 DelayedField_t	497
12.215.1.3 DelayedField	497
12.215.1.4 DFElt	497
12.215.2 Constructor & Destructor Documentation	497
12.215.2.1 MMHelper() [1/5]	497
12.215.2.2 MMHelper() [2/5]	497
12.215.2.3 MMHelper() [3/5]	497
12.215.2.4 MMHelper() [4/5]	497
12.215.2.5 MMHelper() [5/5]	498
12.215.3 Member Function Documentation	498
12.215.3.1 initC()	498
12.215.3.2 initA()	498
12.215.3.3 initB()	498
12.215.3.4 initOut()	498

12.215.3.5 MaxDelayedDim()	498
12.215.3.6 Aunfit()	498
12.215.3.7 Bunfit()	498
12.215.3.8 setOutBounds()	498
12.215.3.9 checkA()	499
12.215.3.10 checkB()	499
12.215.3.11 checkOut() [1/2]	499
12.215.3.12 checkOut() [2/2]	499
12.215.4 Friends And Related Symbol Documentation	499
12.215.4.1 operator<<	499
12.215.5 Field Documentation	499
12.215.5.1 recLevel	499
12.215.5.2 FieldMin	500
12.215.5.3 FieldMax	500
12.215.5.4 Amin	500
12.215.5.5 Amax	500
12.215.5.6 Bmin	500
12.215.5.7 Bmax	500
12.215.5.8 Cmin	500
12.215.5.9 Cmax	500
12.215.5.10 Outmin	500
12.215.5.11 Outmax	500
12.215.5.12 MaxStorableValue	500
12.215.5.13 delayedField	500
12.215.5.14 parseq	501
12.216 MMHelper< FFPACK::RNSInteger< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait > Struct Template Reference	501
12.216.1 Member Typedef Documentation	501
12.216.1.1 Self_t	501
12.216.2 Constructor & Destructor Documentation	501
12.216.2.1 MMHelper() [1/5]	501
12.216.2.2 MMHelper() [2/5]	501
12.216.2.3 MMHelper() [3/5]	502
12.216.2.4 MMHelper() [4/5]	502
12.216.2.5 MMHelper() [5/5]	502
12.216.3 Member Function Documentation	502
12.216.3.1 setNorm()	502
12.216.4 Friends And Related Symbol Documentation	502
12.216.4.1 operator<<	502
12.216.5 Field Documentation	502
12.216.5.1 normA	502
12.216.5.2 normB	502

12.216.5.3 recLevel	502
12.216.5.4 parseq	503
12.217 MMHelper< FFPACK::RNSIntegerMod< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeq← Trait > Struct Template Reference	503
12.217.1 Member Typedef Documentation	503
12.217.1.1 Self_t	503
12.217.2 Constructor & Destructor Documentation	503
12.217.2.1 MMHelper() [1/5]	503
12.217.2.2 MMHelper() [2/5]	503
12.217.2.3 MMHelper() [3/5]	504
12.217.2.4 MMHelper() [4/5]	504
12.217.2.5 MMHelper() [5/5]	504
12.217.3 Member Function Documentation	504
12.217.3.1 setNorm()	504
12.217.4 Friends And Related Symbol Documentation	504
12.217.4.1 operator<<	504
12.217.5 Field Documentation	504
12.217.5.1 normA	504
12.217.5.2 normB	504
12.217.5.3 recLevel	504
12.217.5.4 parseq	505
12.218 MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< Dest >, ParSeqTrait > Struct Tem- plate Reference	505
12.218.1 Member Typedef Documentation	505
12.218.1.1 Self_t	505
12.218.2 Constructor & Destructor Documentation	505
12.218.2.1 MMHelper() [1/4]	505
12.218.2.2 MMHelper() [2/4]	505
12.218.2.3 MMHelper() [3/4]	506
12.218.2.4 MMHelper() [4/4]	506
12.218.3 Friends And Related Symbol Documentation	506
12.218.3.1 operator<<	506
12.218.4 Field Documentation	506
12.218.4.1 recLevel	506
12.218.4.2 parseq	506
12.219 MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeqTrait > Struct Template Reference	506
12.219.1 Member Typedef Documentation	507
12.219.1.1 Self_t	507
12.219.2 Constructor & Destructor Documentation	507
12.219.2.1 MMHelper() [1/5]	507
12.219.2.2 MMHelper() [2/5]	507
12.219.2.3 MMHelper() [3/5]	507

12.219.2.4 MMHelper() [4 / 5]	507
12.219.2.5 MMHelper() [5 / 5]	507
12.219.3 Member Function Documentation	508
12.219.3.1 setNorm()	508
12.219.4 Friends And Related Symbol Documentation	508
12.219.4.1 operator<<	508
12.219.5 Field Documentation	508
12.219.5.1 normA	508
12.219.5.2 normB	508
12.219.5.3 recLevel	508
12.219.5.4 parseq	508
12.220 MMHelper< Field, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait > Struct Template Reference	508
12.220.1 Detailed Description	509
12.220.2 Member Typedef Documentation	509
12.220.2.1 Self_t	509
12.220.3 Constructor & Destructor Documentation	509
12.220.3.1 MMHelper() [1 / 4]	509
12.220.3.2 MMHelper() [2 / 4]	509
12.220.3.3 MMHelper() [3 / 4]	509
12.220.3.4 MMHelper() [4 / 4]	509
12.220.4 Friends And Related Symbol Documentation	509
12.220.4.1 operator<<	509
12.220.5 Field Documentation	510
12.220.5.1 recLevel	510
12.220.5.2 parseq	510
12.221 ModeTraits< Field > Struct Template Reference	510
12.221.1 Detailed Description	510
12.221.2 Member Typedef Documentation	510
12.221.2.1 value	510
12.222 ModeTraits< Givaro::Modular< Element, Compute > > Struct Template Reference	510
12.222.1 Member Typedef Documentation	510
12.222.1.1 value	510
12.223 ModeTraits< Givaro::Modular< Givaro::Integer, Compute > > Struct Template Reference	511
12.223.1 Member Typedef Documentation	511
12.223.1.1 value	511
12.224 ModeTraits< Givaro::Modular< int16_t, Compute > > Struct Template Reference	511
12.224.1 Member Typedef Documentation	511
12.224.1.1 value	511
12.225 ModeTraits< Givaro::Modular< int32_t, Compute > > Struct Template Reference	511
12.225.1 Member Typedef Documentation	511
12.225.1.1 value	511
12.226 ModeTraits< Givaro::Modular< int64_t, uint64_t > > Struct Reference	512

12.226.1 Member Typedef Documentation	512
12.226.1.1 value	512
12.227 ModeTraits< Givaro::Modular< int8_t, Compute > > Struct Template Reference	512
12.227.1 Member Typedef Documentation	512
12.227.1.1 value	512
12.228 ModeTraits< Givaro::Modular< Reclnt::ruint< K >, Compute > > Struct Template Reference	512
12.228.1 Member Typedef Documentation	512
12.228.1.1 value	512
12.229 ModeTraits< Givaro::Modular< uint16_t, Compute > > Struct Template Reference	513
12.229.1 Member Typedef Documentation	513
12.229.1.1 value	513
12.230 ModeTraits< Givaro::Modular< uint32_t, Compute > > Struct Template Reference	513
12.230.1 Member Typedef Documentation	513
12.230.1.1 value	513
12.231 ModeTraits< Givaro::Modular< uint8_t, Compute > > Struct Template Reference	513
12.231.1 Member Typedef Documentation	513
12.231.1.1 value	513
12.232 ModeTraits< Givaro::ModularBalanced< Element > > Struct Template Reference	514
12.232.1 Member Typedef Documentation	514
12.232.1.1 value	514
12.233 ModeTraits< Givaro::ModularBalanced< Givaro::Integer > > Struct Reference	514
12.233.1 Member Typedef Documentation	514
12.233.1.1 value	514
12.234 ModeTraits< Givaro::ModularBalanced< int16_t > > Struct Reference	514
12.234.1 Member Typedef Documentation	514
12.234.1.1 value	514
12.235 ModeTraits< Givaro::ModularBalanced< int32_t > > Struct Reference	515
12.235.1 Member Typedef Documentation	515
12.235.1.1 value	515
12.236 ModeTraits< Givaro::ModularBalanced< int8_t > > Struct Reference	515
12.236.1 Member Typedef Documentation	515
12.236.1.1 value	515
12.237 ModeTraits< Givaro::Montgomery< T > > Struct Template Reference	515
12.237.1 Member Typedef Documentation	515
12.237.1.1 value	515
12.238 ModeTraits< Givaro::ZRing< double > > Struct Reference	515
12.238.1 Member Typedef Documentation	516
12.238.1.1 value	516
12.239 ModeTraits< Givaro::ZRing< float > > Struct Reference	516
12.239.1 Member Typedef Documentation	516
12.239.1.1 value	516
12.240 ModeTraits< Givaro::ZRing< Givaro::Integer > > Struct Reference	516

12.240.1 Member Typedef Documentation	516
12.240.1.1 value	516
12.241 ModularBalanced< T > Class Template Reference	516
12.242 ModularTag Struct Reference	516
12.242.1 Detailed Description	517
12.243 Montgomery< T > Class Template Reference	517
12.244 need_field_characteristic< Field > Struct Template Reference	517
12.244.1 Field Documentation	517
12.244.1.1 value	517
12.245 need_field_characteristic< Givaro::Modular< Field > > Struct Template Reference	517
12.245.1 Field Documentation	517
12.245.1.1 value	517
12.246 need_field_characteristic< Givaro::ModularBalanced< Field > > Struct Template Reference	517
12.246.1 Field Documentation	517
12.246.1.1 value	517
12.247 NoSimd< T > Struct Template Reference	518
12.247.1 Member Typedef Documentation	518
12.247.1.1 vect_t	518
12.247.1.2 scalar_t	518
12.247.1.3 aligned_allocator	518
12.247.1.4 aligned_vector	518
12.247.1.5 is_same_element	518
12.247.2 Member Function Documentation	519
12.247.2.1 type_string()	519
12.247.2.2 valid()	519
12.247.2.3 compliant()	519
12.247.3 Field Documentation	519
12.247.3.1 vect_size	519
12.247.3.2 alignment	519
12.248 Parallel< C, P > Struct Template Reference	519
12.248.1 Member Typedef Documentation	519
12.248.1.1 Cut	519
12.248.1.2 Param	520
12.248.2 Constructor & Destructor Documentation	520
12.248.2.1 Parallel()	520
12.248.3 Member Function Documentation	520
12.248.3.1 numthreads()	520
12.248.3.2 set_numthreads()	520
12.248.4 Friends And Related Symbol Documentation	520
12.248.4.1 operator<<	520
12.249 RNSInteger< RNS >::RandIter Class Reference	520
12.249.1 Constructor & Destructor Documentation	521

12.249.1.1 RandIter()	521
12.249.2 Member Function Documentation	521
12.249.2.1 random() [1/2]	521
12.249.2.2 random() [2/2]	521
12.249.2.3 operator>() [1/2]	521
12.249.2.4 operator>() [2/2]	521
12.249.2.5 ring()	521
12.250 RNSIntegerMod< RNS >::RandIter Class Reference	521
12.250.1 Constructor & Destructor Documentation	522
12.250.1.1 RandIter()	522
12.250.2 Member Function Documentation	522
12.250.2.1 random() [1/2]	522
12.250.2.2 random() [2/2]	522
12.250.2.3 operator>() [1/2]	522
12.250.2.4 operator>() [2/2]	522
12.250.2.5 ring()	522
12.251 readMyMachineType< Field, T > Struct Template Reference	522
12.251.1 Member Typedef Documentation	523
12.251.1.1 Element	523
12.251.1.2 Element_ptr	523
12.251.2 Member Function Documentation	523
12.251.2.1 operator>()	523
12.252 readMyMachineType< Field, mpz_t > Struct Template Reference	523
12.252.1 Member Typedef Documentation	523
12.252.1.1 Element	523
12.252.1.2 Element_ptr	523
12.252.2 Member Function Documentation	524
12.252.2.1 operator>()	524
12.253 Recursive Struct Reference	524
12.254 Recursive Struct Reference	524
12.255 rint< K > Class Template Reference	524
12.256 rns_double Struct Reference	524
12.256.1 Member Typedef Documentation	525
12.256.1.1 integer	525
12.256.1.2 ModField	525
12.256.1.3 BasisElement	525
12.256.1.4 Element	525
12.256.1.5 Element_ptr	525
12.256.1.6 ConstElement_ptr	526
12.256.2 Constructor & Destructor Documentation	526
12.256.2.1 rns_double() [1/4]	526
12.256.2.2 rns_double() [2/4]	526

12.256.2.3 rns_double() [3/4]	526
12.256.2.4 rns_double() [4/4]	526
12.256.3 Member Function Documentation	526
12.256.3.1 precompute_cst()	526
12.256.3.2 init() [1/3]	526
12.256.3.3 init() [2/3]	526
12.256.3.4 init_transpose()	527
12.256.3.5 convert() [1/2]	527
12.256.3.6 convert_transpose()	527
12.256.3.7 reduce()	527
12.256.3.8 init() [3/3]	527
12.256.3.9 convert() [2/2]	528
12.256.4 Field Documentation	528
12.256.4.1 _basis	528
12.256.4.2 _basisMax	528
12.256.4.3 _negbasis	528
12.256.4.4 _invbasis	528
12.256.4.5 _field_rns	528
12.256.4.6 _M	528
12.256.4.7 _Mi	528
12.256.4.8 _MMi	528
12.256.4.9 _crt_in	528
12.256.4.10 _crt_out	528
12.256.4.11 _size	529
12.256.4.12 _pbits	529
12.256.4.13 _ldm	529
12.256.4.14 _mi_sum	529
12.257 rns_double_elt Struct Reference	529
12.257.1 Constructor & Destructor Documentation	529
12.257.1.1 rns_double_elt() [1/3]	529
12.257.1.2 ~rns_double_elt()	529
12.257.1.3 rns_double_elt() [2/3]	530
12.257.1.4 rns_double_elt() [3/3]	530
12.257.2 Member Function Documentation	530
12.257.2.1 operator&() [1/2]	530
12.257.2.2 operator&() [2/2]	530
12.257.3 Field Documentation	530
12.257.3.1 _ptr	530
12.257.3.2 _stride	530
12.257.3.3 _alloc	530
12.258 rns_double_elt_cstptr Struct Reference	530
12.258.1 Constructor & Destructor Documentation	531

12.258.1.1 rns_double_elt_cstptr() [1/5]	531
12.258.1.2 rns_double_elt_cstptr() [2/5]	531
12.258.1.3 rns_double_elt_cstptr() [3/5]	531
12.258.1.4 rns_double_elt_cstptr() [4/5]	531
12.258.1.5 rns_double_elt_cstptr() [5/5]	531
12.258.2 Member Function Documentation	531
12.258.2.1 operator&() [1/2]	531
12.258.2.2 operator*() [1/2]	531
12.258.2.3 operator[]() [1/2]	532
12.258.2.4 operator[]() [2/2]	532
12.258.2.5 operator++()	532
12.258.2.6 operator--()	532
12.258.2.7 operator+()	532
12.258.2.8 operator-()	532
12.258.2.9 operator+=()	532
12.258.2.10 operator-=()	532
12.258.2.11 operator=()	532
12.258.2.12 operator<()	532
12.258.2.13 operator"!=(())	532
12.258.2.14 operator&() [2/2]	532
12.258.3 Field Documentation	533
12.258.3.1 other	533
12.258.3.2 _ptr	533
12.258.3.3 _stride	533
12.258.3.4 _alloc	533
12.259 rns_double_elt_ptr Struct Reference	533
12.259.1 Constructor & Destructor Documentation	534
12.259.1.1 rns_double_elt_ptr() [1/5]	534
12.259.1.2 rns_double_elt_ptr() [2/5]	534
12.259.1.3 rns_double_elt_ptr() [3/5]	534
12.259.1.4 rns_double_elt_ptr() [4/5]	534
12.259.1.5 rns_double_elt_ptr() [5/5]	534
12.259.2 Member Function Documentation	534
12.259.2.1 operator&() [1/2]	534
12.259.2.2 operator*() [1/2]	534
12.259.2.3 operator[]() [1/2]	534
12.259.2.4 operator[]() [2/2]	534
12.259.2.5 operator++()	534
12.259.2.6 operator--()	534
12.259.2.7 operator+()	535
12.259.2.8 operator-()	535
12.259.2.9 operator+=()	535

12.259.2.10 operator==()	535
12.259.2.11 operator=()	535
12.259.2.12 operator<()	535
12.259.2.13 operator"!=()	535
12.259.2.14 operator&() [2/2]	535
12.259.3 Field Documentation	535
12.259.3.1 other	535
12.259.3.2 _ptr	535
12.259.3.3 _stride	535
12.259.3.4 _alloc	535
12.260 rns_double_extended Struct Reference	536
12.260.1 Member Typedef Documentation	536
12.260.1.1 integer	536
12.260.1.2 ModField	536
12.260.1.3 BasisElement	537
12.260.1.4 Element	537
12.260.1.5 Element_ptr	537
12.260.1.6 ConstElement_ptr	537
12.260.2 Constructor & Destructor Documentation	537
12.260.2.1 rns_double_extended() [1/3]	537
12.260.2.2 rns_double_extended() [2/3]	537
12.260.2.3 rns_double_extended() [3/3]	537
12.260.3 Member Function Documentation	537
12.260.3.1 precompute_cst()	537
12.260.3.2 init() [1/3]	537
12.260.3.3 init() [2/3]	538
12.260.3.4 convert() [1/2]	538
12.260.3.5 init() [3/3]	538
12.260.3.6 convert() [2/2]	538
12.260.3.7 reduce()	538
12.260.4 Field Documentation	538
12.260.4.1 _basis	538
12.260.4.2 _basisMax	538
12.260.4.3 _negbasis	538
12.260.4.4 _invbasis	539
12.260.4.5 _field_rns	539
12.260.4.6 _M	539
12.260.4.7 _Mi	539
12.260.4.8 _MMi	539
12.260.4.9 _crt_in	539
12.260.4.10 _crt_out	539
12.260.4.11 _size	539

12.260.4.12 _pbits	539
12.260.4.13 _ldm	539
12.261 RNSElementTag Struct Reference	539
12.261.1 Detailed Description	539
12.262 RNSInteger< RNS > Class Template Reference	539
12.262.1 Member Typedef Documentation	540
12.262.1.1 BasisElement	540
12.262.1.2 integer	540
12.262.1.3 Element	541
12.262.1.4 Element_ptr	541
12.262.1.5 ConstElement_ptr	541
12.262.2 Constructor & Destructor Documentation	541
12.262.2.1 RNSInteger() [1/2]	541
12.262.2.2 RNSInteger() [2/2]	541
12.262.3 Member Function Documentation	541
12.262.3.1 rns()	541
12.262.3.2 size()	541
12.262.3.3 isOne()	541
12.262.3.4 isMOne()	541
12.262.3.5 isZero()	541
12.262.3.6 characteristic()	542
12.262.3.7 cardinality()	542
12.262.3.8 init() [1/2]	542
12.262.3.9 init() [2/2]	542
12.262.3.10 reduce() [1/2]	542
12.262.3.11 reduce() [2/2]	542
12.262.3.12 convert()	542
12.262.3.13 assign()	542
12.262.3.14 write() [1/2]	542
12.262.3.15 write() [2/2]	543
12.262.4 Field Documentation	543
12.262.4.1 _rns	543
12.262.4.2 one	543
12.262.4.3 mOne	543
12.262.4.4 zero	543
12.263 RNSIntegerMod< RNS > Class Template Reference	543
12.263.1 Member Typedef Documentation	544
12.263.1.1 Element	544
12.263.1.2 Element_ptr	544
12.263.1.3 ConstElement_ptr	545
12.263.1.4 BasisElement	545
12.263.1.5 ModField	545

12.263.1.6 integer	545
12.263.2 Constructor & Destructor Documentation	545
12.263.2.1 RNSIntegerMod()	545
12.263.3 Member Function Documentation	545
12.263.3.1 rns()	545
12.263.3.2 delayed()	545
12.263.3.3 size()	545
12.263.3.4 isOne()	545
12.263.3.5 isMOne()	545
12.263.3.6 isZero()	546
12.263.3.7 characteristic() [1/2]	546
12.263.3.8 characteristic() [2/2]	546
12.263.3.9 cardinality() [1/2]	546
12.263.3.10 cardinality() [2/2]	546
12.263.3.11 minElement()	546
12.263.3.12 maxElement()	546
12.263.3.13 init() [1/3]	546
12.263.3.14 init() [2/3]	546
12.263.3.15 reduce() [1/2]	546
12.263.3.16 reduce() [2/2]	547
12.263.3.17 init() [3/3]	547
12.263.3.18 convert()	547
12.263.3.19 assign()	547
12.263.3.20 add()	547
12.263.3.21 sub()	547
12.263.3.22 neg()	547
12.263.3.23 mul()	547
12.263.3.24 axpyin()	548
12.263.3.25 inv()	548
12.263.3.26 areEqual()	548
12.263.3.27 write() [1/2]	548
12.263.3.28 write() [2/2]	548
12.263.3.29 reduce_modp() [1/2]	548
12.263.3.30 write_matrix()	548
12.263.3.31 write_matrix_long()	548
12.263.3.32 reduce_modp() [2/2]	549
12.263.3.33 reduce_modp_rnsmajor()	549
12.263.4 Field Documentation	549
12.263.4.1 _p	549
12.263.4.2 _Mi_modp_rns	549
12.263.4.3 _iM_modp_rns	549
12.263.4.4 _rns	549

12.263.4.5 _F	549
12.263.4.6 _RNSdelayed	549
12.263.4.7 one	549
12.263.4.8 mOne	550
12.263.4.9 zero	550
12.264 rnsRandIter< RNS > Class Template Reference	550
12.264.1 Constructor & Destructor Documentation	550
12.264.1.1 rnsRandIter()	550
12.264.2 Member Function Documentation	550
12.264.2.1 random() [1/2]	550
12.264.2.2 operator>() [1/2]	551
12.264.2.3 operator>() [2/2]	551
12.264.2.4 random() [2/2]	551
12.264.2.5 ring()	551
12.265 Row Struct Reference	551
12.266 ruint< K > Class Template Reference	551
12.267 ScalFunctions< Element > Struct Template Reference	551
12.267.1 Member Typedef Documentation	552
12.267.1.1 vectElt	552
12.267.2 Member Function Documentation	552
12.267.2.1 genInputs()	552
12.267.2.2 genInputsWithZero()	552
12.267.2.3 zero()	552
12.267.2.4 vand()	553
12.267.2.5 vor()	553
12.267.2.6 vxor()	553
12.267.2.7 vandnot()	553
12.267.2.8 add()	553
12.267.2.9 addin()	553
12.267.2.10 sub()	553
12.267.2.11 subin()	553
12.267.2.12 mul()	553
12.267.2.13 mulin()	554
12.267.2.14 div()	554
12.267.2.15 fmadd()	554
12.267.2.16 fmaddin()	554
12.267.2.17 fmsub()	554
12.267.2.18 fmsubin()	554
12.267.2.19 fnmadd()	554
12.267.2.20 fnmaddin()	555
12.267.2.21 lesser()	555
12.267.2.22 lesser_eq()	555

12.267.2.23 greater()	555
12.267.2.24 greater_eq()	555
12.267.2.25 eq()	555
12.267.2.26 unpacklo()	555
12.267.2.27 unpackhi()	555
12.267.2.28 unpacklohi()	556
12.267.2.29 pack_even()	556
12.267.2.30 pack_odd()	556
12.267.2.31 pack()	556
12.267.2.32 blend()	556
12.268 ScalFunctionsBase< Element, Enable > Struct Template Reference	556
12.269 ScalFunctionsBase< Element, typename enable_if< is_floating_point< Element >::value >::type > Struct Template Reference	557
12.269.1 Member Function Documentation	557
12.269.1.1 get_default_random_generator()	557
12.269.1.2 ceil()	557
12.269.1.3 floor()	557
12.269.1.4 round()	557
12.269.1.5 blendv()	557
12.269.1.6 fma()	558
12.269.2 Field Documentation	558
12.269.2.1 _zero	558
12.269.2.2 cmp_true	558
12.269.2.3 cmp_false	558
12.270 ScalFunctionsBase< Element, typename enable_if< is_integral< Element >::value >::type > Struct Template Reference	558
12.270.1 Member Function Documentation	559
12.270.1.1 get_default_random_generator()	559
12.270.1.2 round()	559
12.270.1.3 fma()	559
12.270.1.4 mullo()	559
12.270.1.5 mulhi()	559
12.270.1.6 mulx()	559
12.270.1.7 fmaddx()	559
12.270.1.8 fmaddxin()	559
12.270.1.9 fmsubx()	560
12.270.1.10 fmsubxin()	560
12.270.1.11 fnmaddx()	560
12.270.1.12 fnmaddxin()	560
12.270.1.13 sra()	560
12.270.1.14 srl()	560
12.270.1.15 sll()	560
12.270.2 Field Documentation	560

12.270.2.1 <code>_zero</code>	560
12.270.2.2 <code>cmp_true</code>	561
12.270.2.3 <code>cmp_false</code>	561
12.271 Sequential Struct Reference	561
12.271.1 Constructor & Destructor Documentation	561
12.271.1.1 <code>Sequential()</code> [1/3]	561
12.271.1.2 <code>Sequential()</code> [2/3]	561
12.271.1.3 <code>Sequential()</code> [3/3]	561
12.271.2 Member Function Documentation	561
12.271.2.1 <code>numthreads()</code>	561
12.271.3 Friends And Related Symbol Documentation	561
12.271.3.1 <code>operator<<</code>	561
12.272 <code>Simd128_impl< ArithType, Int, Signed, Size ></code> Struct Template Reference	562
12.273 <code>Simd128_impl< true, false, true, 4 ></code> Struct Reference	562
12.274 <code>Simd128_impl< true, false, true, 8 ></code> Struct Reference	562
12.275 <code>Simd128_impl< true, true, false, 2 ></code> Struct Reference	562
12.275.1 Member Typedef Documentation	564
12.275.1.1 <code>scalar_t</code>	564
12.275.1.2 <code>aligned_allocator</code>	564
12.275.1.3 <code>aligned_vector</code>	564
12.275.1.4 <code>is_same_element</code>	564
12.275.1.5 <code>vect_t</code>	564
12.275.2 Member Function Documentation	564
12.275.2.1 <code>type_string()</code>	564
12.275.2.2 <code>set1()</code> [1/2]	564
12.275.2.3 <code>set()</code> [1/2]	565
12.275.2.4 <code>gather()</code> [1/2]	565
12.275.2.5 <code>load()</code> [1/2]	565
12.275.2.6 <code>loadu()</code> [1/2]	565
12.275.2.7 <code>store()</code> [1/2]	565
12.275.2.8 <code>storeu()</code> [1/2]	565
12.275.2.9 <code>stream()</code> [1/2]	565
12.275.2.10 <code>sra()</code>	565
12.275.2.11 <code>greater()</code>	565
12.275.2.12 <code>lesser()</code>	566
12.275.2.13 <code>greater_eq()</code>	566
12.275.2.14 <code>lesser_eq()</code>	566
12.275.2.15 <code>mulhi()</code>	566
12.275.2.16 <code>mulx()</code>	566
12.275.2.17 <code>fmaddx()</code>	566
12.275.2.18 <code>fmaddxin()</code>	566
12.275.2.19 <code>fnmaddx()</code>	566

12.275.2.20 fmaddxin()	566
12.275.2.21 fmsubx()	567
12.275.2.22 fmsubxin()	567
12.275.2.23 hadd_to_scal()	567
12.275.2.24 valid()	567
12.275.2.25 compliant()	567
12.275.2.26 set1() [2/2]	567
12.275.2.27 set() [2/2]	567
12.275.2.28 gather() [2/2]	567
12.275.2.29 load() [2/2]	567
12.275.2.30 loadu() [2/2]	568
12.275.2.31 store() [2/2]	568
12.275.2.32 storeu() [2/2]	568
12.275.2.33 stream() [2/2]	568
12.275.2.34 sll()	568
12.275.2.35 srl()	568
12.275.2.36 shuffle()	568
12.275.2.37 unpacklo_intrinsic()	568
12.275.2.38 unpackhi_intrinsic()	568
12.275.2.39 unpacklo()	568
12.275.2.40 unpackhi()	569
12.275.2.41 unpacklohi()	569
12.275.2.42 pack_even()	569
12.275.2.43 pack_odd()	569
12.275.2.44 pack()	569
12.275.2.45 transpose()	569
12.275.2.46 blend()	569
12.275.2.47 add()	569
12.275.2.48 addin()	570
12.275.2.49 sub()	570
12.275.2.50 subin()	570
12.275.2.51 mullo()	570
12.275.2.52 mul()	570
12.275.2.53 fmadd()	570
12.275.2.54 fmaddin()	570
12.275.2.55 fnmadd()	570
12.275.2.56 fnmaddin()	570
12.275.2.57 fmsub()	571
12.275.2.58 fmsubin()	571
12.275.2.59 eq()	571
12.275.2.60 round()	571
12.275.2.61 mod()	571

12.275.2.62 zero()	571
12.275.2.63 sll128()	571
12.275.2.64 srl128()	571
12.275.2.65 vand()	571
12.275.2.66 vor()	572
12.275.2.67 vxor()	572
12.275.2.68 vandnot()	572
12.275.3 Field Documentation	572
12.275.3.1 vect_size	572
12.275.3.2 alignment	572
12.276 Simd128_impl< true, true, false, 4 > Struct Reference	572
12.276.1 Member Typedef Documentation	574
12.276.1.1 scalar_t	574
12.276.1.2 aligned_allocator	574
12.276.1.3 aligned_vector	574
12.276.1.4 is_same_element	574
12.276.1.5 vect_t	574
12.276.2 Member Function Documentation	575
12.276.2.1 type_string()	575
12.276.2.2 set1() [1/2]	575
12.276.2.3 set() [1/2]	575
12.276.2.4 gather() [1/2]	575
12.276.2.5 load() [1/2]	575
12.276.2.6 loadu() [1/2]	575
12.276.2.7 store() [1/2]	575
12.276.2.8 storeu() [1/2]	575
12.276.2.9 stream() [1/2]	575
12.276.2.10 sra()	575
12.276.2.11 greater()	576
12.276.2.12 lesser()	576
12.276.2.13 greater_eq()	576
12.276.2.14 lesser_eq()	576
12.276.2.15 mulhi()	576
12.276.2.16 mulx()	576
12.276.2.17 fmaddx()	576
12.276.2.18 fmaddxin()	576
12.276.2.19 fnmaddx()	576
12.276.2.20 fnmaddxin()	577
12.276.2.21 fmsubx()	577
12.276.2.22 fmsubxin()	577
12.276.2.23 hadd_to_scal()	577
12.276.2.24 valid()	577

12.276.2.25 compliant()	577
12.276.2.26 set1() [2/2]	577
12.276.2.27 set() [2/2]	577
12.276.2.28 gather() [2/2]	577
12.276.2.29 load() [2/2]	578
12.276.2.30 loadu() [2/2]	578
12.276.2.31 store() [2/2]	578
12.276.2.32 storeu() [2/2]	578
12.276.2.33 stream() [2/2]	578
12.276.2.34 sll()	578
12.276.2.35 srl()	578
12.276.2.36 shuffle()	578
12.276.2.37 unpacklo_intrinsic()	578
12.276.2.38 unpackhi_intrinsic()	578
12.276.2.39 unpacklo()	579
12.276.2.40 unpackhi()	579
12.276.2.41 unpacklohi()	579
12.276.2.42 pack_even()	579
12.276.2.43 pack_odd()	579
12.276.2.44 pack()	579
12.276.2.45 transpose()	579
12.276.2.46 blend()	579
12.276.2.47 add()	579
12.276.2.48 addin()	580
12.276.2.49 sub()	580
12.276.2.50 subin()	580
12.276.2.51 mullo()	580
12.276.2.52 mul()	580
12.276.2.53 fmadd()	580
12.276.2.54 fmaddin()	580
12.276.2.55 fnmadd()	580
12.276.2.56 fnmaddin()	580
12.276.2.57 fmsub()	581
12.276.2.58 fmsubin()	581
12.276.2.59 eq()	581
12.276.2.60 round()	581
12.276.2.61 mod()	581
12.276.2.62 zero()	581
12.276.2.63 sll128()	581
12.276.2.64 srl128()	581
12.276.2.65 vand()	581
12.276.2.66 vor()	582

12.276.2.67 vxor()	582
12.276.2.68 vandnot()	582
12.276.3 Field Documentation	582
12.276.3.1 vect_size	582
12.276.3.2 alignment	582
12.277 Simd128_impl< true, true, false, 8 > Struct Reference	582
12.277.1 Member Typedef Documentation	584
12.277.1.1 scalar_t	584
12.277.1.2 aligned_allocator	584
12.277.1.3 aligned_vector	584
12.277.1.4 is_same_element	584
12.277.1.5 vect_t	585
12.277.2 Member Function Documentation	585
12.277.2.1 type_string()	585
12.277.2.2 set1() [1/2]	585
12.277.2.3 set() [1/2]	585
12.277.2.4 gather() [1/2]	585
12.277.2.5 load() [1/2]	585
12.277.2.6 loadu() [1/2]	585
12.277.2.7 store() [1/2]	585
12.277.2.8 storeu() [1/2]	585
12.277.2.9 stream() [1/2]	585
12.277.2.10 sra()	585
12.277.2.11 greater()	586
12.277.2.12 lesser()	586
12.277.2.13 greater_eq()	586
12.277.2.14 lesser_eq()	586
12.277.2.15 mullo()	586
12.277.2.16 mulhi()	586
12.277.2.17 mulx()	586
12.277.2.18 fmaddx()	586
12.277.2.19 fmaddxin()	586
12.277.2.20 fnmaddx()	587
12.277.2.21 fnmaddxin()	587
12.277.2.22 fmsubx()	587
12.277.2.23 fmsubxin()	587
12.277.2.24 hadd_to_scal()	587
12.277.2.25 valid()	587
12.277.2.26 compliant()	587
12.277.2.27 set1() [2/2]	587
12.277.2.28 set() [2/2]	587
12.277.2.29 gather() [2/2]	588

12.277.2.30 get()	588
12.277.2.31 load() [2/2]	588
12.277.2.32 loadu() [2/2]	588
12.277.2.33 store() [2/2]	588
12.277.2.34 storeu() [2/2]	588
12.277.2.35 stream() [2/2]	588
12.277.2.36 sll()	588
12.277.2.37 srl()	588
12.277.2.38 shuffle()	588
12.277.2.39 unpacklo_intrinsic()	589
12.277.2.40 unpackhi_intrinsic()	589
12.277.2.41 unpacklo()	589
12.277.2.42 unpackhi()	589
12.277.2.43 unpacklohi()	589
12.277.2.44 pack_even()	589
12.277.2.45 pack_odd()	589
12.277.2.46 pack()	589
12.277.2.47 transpose()	589
12.277.2.48 blend()	590
12.277.2.49 add()	590
12.277.2.50 addin()	590
12.277.2.51 sub()	590
12.277.2.52 subin()	590
12.277.2.53 mul()	590
12.277.2.54 fmadd()	590
12.277.2.55 fmaddin()	590
12.277.2.56 fnmadd()	590
12.277.2.57 fnmaddin()	591
12.277.2.58 fmsub()	591
12.277.2.59 fmsubin()	591
12.277.2.60 eq()	591
12.277.2.61 round()	591
12.277.2.62 mask_high()	591
12.277.2.63 mulhi_fast()	591
12.277.2.64 mod()	591
12.277.2.65 signbits()	591
12.277.2.66 zero()	592
12.277.2.67 sll128()	592
12.277.2.68 srl128()	592
12.277.2.69 vand()	592
12.277.2.70 vor()	592
12.277.2.71 vxor()	592

12.277.2.72 vandnot()	592
12.277.3 Field Documentation	592
12.277.3.1 vect_size	592
12.277.3.2 alignment	592
12.278 Simd128_impl< true, true, true, 2 > Struct Reference	592
12.278.1 Member Typedef Documentation	594
12.278.1.1 vect_t	594
12.278.1.2 scalar_t	594
12.278.1.3 aligned_allocator	595
12.278.1.4 aligned_vector	595
12.278.1.5 is_same_element	595
12.278.2 Member Function Documentation	595
12.278.2.1 type_string()	595
12.278.2.2 valid()	595
12.278.2.3 compliant()	595
12.278.2.4 set1()	595
12.278.2.5 set()	595
12.278.2.6 gather()	595
12.278.2.7 load()	595
12.278.2.8 loadu()	596
12.278.2.9 store()	596
12.278.2.10 storeu()	596
12.278.2.11 stream()	596
12.278.2.12 sll()	596
12.278.2.13 srl()	596
12.278.2.14 sra()	596
12.278.2.15 shuffle()	596
12.278.2.16 unpacklo_intrinsic()	596
12.278.2.17 unpackhi_intrinsic()	596
12.278.2.18 unpacklo()	597
12.278.2.19 unpackhi()	597
12.278.2.20 unpacklohi()	597
12.278.2.21 pack_even()	597
12.278.2.22 pack_odd()	597
12.278.2.23 pack()	597
12.278.2.24 transpose()	597
12.278.2.25 blend()	597
12.278.2.26 add()	598
12.278.2.27 addin()	598
12.278.2.28 sub()	598
12.278.2.29 subin()	598
12.278.2.30 mullo()	598

12.278.2.31 mul()	598
12.278.2.32 mulhi()	598
12.278.2.33 mulx()	598
12.278.2.34 fmadd()	598
12.278.2.35 fmaddin()	598
12.278.2.36 fmaddx()	599
12.278.2.37 fmaddxin()	599
12.278.2.38 fnmadd()	599
12.278.2.39 fnmaddin()	599
12.278.2.40 fnmaddx()	599
12.278.2.41 fnmaddxin()	599
12.278.2.42 fmsub()	599
12.278.2.43 fmsubin()	599
12.278.2.44 fmsubx()	599
12.278.2.45 fmsubxin()	600
12.278.2.46 eq()	600
12.278.2.47 greater()	600
12.278.2.48 lesser()	600
12.278.2.49 greater_eq()	600
12.278.2.50 lesser_eq()	600
12.278.2.51 hadd_to_scal()	600
12.278.2.52 round()	600
12.278.2.53 mod()	600
12.278.2.54 zero()	601
12.278.2.55 sll128()	601
12.278.2.56 srl128()	601
12.278.2.57 vand()	601
12.278.2.58 vor()	601
12.278.2.59 vxor()	601
12.278.2.60 vandnot()	601
12.278.3 Field Documentation	601
12.278.3.1 vect_size	601
12.278.3.2 alignment	601
12.279 Simd128_impl< true, true, true, 4 > Struct Reference	602
12.279.1 Member Typedef Documentation	603
12.279.1.1 vect_t	603
12.279.1.2 scalar_t	603
12.279.1.3 aligned_allocator	604
12.279.1.4 aligned_vector	604
12.279.1.5 is_same_element	604
12.279.2 Member Function Documentation	604
12.279.2.1 type_string()	604

12.279.2.2 valid()	604
12.279.2.3 compliant()	604
12.279.2.4 set1()	604
12.279.2.5 set()	604
12.279.2.6 gather()	604
12.279.2.7 load()	604
12.279.2.8 loadu()	604
12.279.2.9 store()	605
12.279.2.10 storeu()	605
12.279.2.11 stream()	605
12.279.2.12 sll()	605
12.279.2.13 srl()	605
12.279.2.14 sra()	605
12.279.2.15 shuffle()	605
12.279.2.16 unpacklo_intrinsic()	605
12.279.2.17 unpackhi_intrinsic()	605
12.279.2.18 unpacklo()	605
12.279.2.19 unpackhi()	606
12.279.2.20 unpacklohi()	606
12.279.2.21 pack_even()	606
12.279.2.22 pack_odd()	606
12.279.2.23 pack()	606
12.279.2.24 transpose()	606
12.279.2.25 blend()	606
12.279.2.26 add()	606
12.279.2.27 addin()	606
12.279.2.28 sub()	607
12.279.2.29 subin()	607
12.279.2.30 mullo()	607
12.279.2.31 mul()	607
12.279.2.32 mulhi()	607
12.279.2.33 mulx()	607
12.279.2.34 fmadd()	607
12.279.2.35 fmaddin()	607
12.279.2.36 fmaddx()	607
12.279.2.37 fmaddxin()	608
12.279.2.38 fnmadd()	608
12.279.2.39 fnmaddin()	608
12.279.2.40 fnmaddx()	608
12.279.2.41 fnmaddxin()	608
12.279.2.42 fmsub()	608
12.279.2.43 fmsubin()	608

12.279.2.44 fmsubx()	608
12.279.2.45 fmsubxin()	608
12.279.2.46 eq()	609
12.279.2.47 greater()	609
12.279.2.48 lesser()	609
12.279.2.49 greater_eq()	609
12.279.2.50 lesser_eq()	609
12.279.2.51 hadd_to_scal()	609
12.279.2.52 round()	609
12.279.2.53 mod()	609
12.279.2.54 zero()	609
12.279.2.55 sll128()	610
12.279.2.56 srl128()	610
12.279.2.57 vand()	610
12.279.2.58 vor()	610
12.279.2.59 vxor()	610
12.279.2.60 vandnot()	610
12.279.3 Field Documentation	610
12.279.3.1 vect_size	610
12.279.3.2 alignment	610
12.280 Simd128_impl< true, true, true, 8 > Struct Reference	610
12.280.1 Member Typedef Documentation	612
12.280.1.1 vect_t	612
12.280.1.2 scalar_t	613
12.280.1.3 aligned_allocator	613
12.280.1.4 aligned_vector	613
12.280.1.5 is_same_element	613
12.280.2 Member Function Documentation	613
12.280.2.1 type_string()	613
12.280.2.2 valid()	613
12.280.2.3 compliant()	613
12.280.2.4 set1()	613
12.280.2.5 set()	613
12.280.2.6 gather()	613
12.280.2.7 get()	613
12.280.2.8 load()	614
12.280.2.9 loadu()	614
12.280.2.10 store()	614
12.280.2.11 storeu()	614
12.280.2.12 stream()	614
12.280.2.13 sll()	614
12.280.2.14 srl()	614

12.280.2.15 sra()	614
12.280.2.16 shuffle()	614
12.280.2.17 unpacklo_intrinsic()	614
12.280.2.18 unpackhi_intrinsic()	615
12.280.2.19 unpacklo()	615
12.280.2.20 unpackhi()	615
12.280.2.21 unpacklohi()	615
12.280.2.22 pack_even()	615
12.280.2.23 pack_odd()	615
12.280.2.24 pack()	615
12.280.2.25 transpose()	615
12.280.2.26 blend()	615
12.280.2.27 add()	616
12.280.2.28 addin()	616
12.280.2.29 sub()	616
12.280.2.30 subin()	616
12.280.2.31 mullo()	616
12.280.2.32 mul()	616
12.280.2.33 mulhi()	616
12.280.2.34 mulx()	616
12.280.2.35 fmadd()	616
12.280.2.36 fmaddin()	616
12.280.2.37 fmaddx()	617
12.280.2.38 fmaddxin()	617
12.280.2.39 fnmadd()	617
12.280.2.40 fnmaddin()	617
12.280.2.41 fnmaddx()	617
12.280.2.42 fnmaddxin()	617
12.280.2.43 fmsub()	617
12.280.2.44 fmsubin()	617
12.280.2.45 fmsubx()	617
12.280.2.46 fmsubxin()	618
12.280.2.47 eq()	618
12.280.2.48 greater()	618
12.280.2.49 lesser()	618
12.280.2.50 greater_eq()	618
12.280.2.51 lesser_eq()	618
12.280.2.52 hadd_to_scal()	618
12.280.2.53 round()	618
12.280.2.54 mask_high()	618
12.280.2.55 mulhi_fast()	618
12.280.2.56 mod()	619

12.280.2.57 signbits()	619
12.280.2.58 zero()	619
12.280.2.59 sll128()	619
12.280.2.60 srl128()	619
12.280.2.61 vand()	619
12.280.2.62 vor()	619
12.280.2.63 vxor()	619
12.280.2.64 vandnot()	619
12.280.3 Field Documentation	620
12.280.3.1 vect_size	620
12.280.3.2 alignment	620
12.281 Simd128i_base Struct Reference	620
12.281.1 Member Typedef Documentation	620
12.281.1.1 vect_t	620
12.281.2 Member Function Documentation	620
12.281.2.1 zero()	620
12.281.2.2 sll128()	620
12.281.2.3 srl128()	621
12.281.2.4 vand()	621
12.281.2.5 vor()	621
12.281.2.6 vxor()	621
12.281.2.7 vandnot()	621
12.282 Simd256_impl< ArithType, Int, Signed, Size > Struct Template Reference	621
12.283 Simd256_impl< true, false, true, 4 > Struct Reference	621
12.284 Simd256_impl< true, false, true, 8 > Struct Reference	622
12.284.1 Member Typedef Documentation	623
12.284.1.1 vect_t	623
12.284.1.2 scalar_t	623
12.284.1.3 aligned_allocator	623
12.284.1.4 aligned_vector	623
12.284.1.5 is_same_element	623
12.284.2 Member Function Documentation	623
12.284.2.1 type_string()	623
12.284.2.2 valid()	624
12.284.2.3 compliant()	624
12.284.2.4 zero()	624
12.284.2.5 set1()	624
12.284.2.6 set()	624
12.284.2.7 gather()	624
12.284.2.8 load()	624
12.284.2.9 loadu()	624
12.284.2.10 store()	624

12.284.2.11 storeu()	624
12.284.2.12 stream()	625
12.284.2.13 unpacklo_intrinsic()	625
12.284.2.14 unpackhi_intrinsic()	625
12.284.2.15 unpacklo()	625
12.284.2.16 unpackhi()	625
12.284.2.17 unpacklohi()	625
12.284.2.18 pack_even()	625
12.284.2.19 pack_odd()	625
12.284.2.20 pack()	625
12.284.2.21 transpose()	626
12.284.2.22 blend()	626
12.284.2.23 blendv()	626
12.284.2.24 add()	626
12.284.2.25 addin()	626
12.284.2.26 sub()	626
12.284.2.27 subin()	626
12.284.2.28 mul()	626
12.284.2.29 mulin()	626
12.284.2.30 div()	627
12.284.2.31 fmadd()	627
12.284.2.32 fmaddin()	627
12.284.2.33 fnmadd()	627
12.284.2.34 fnmaddin()	627
12.284.2.35 fmsub()	627
12.284.2.36 fmsubin()	627
12.284.2.37 eq()	627
12.284.2.38 lesser()	627
12.284.2.39 lesser_eq()	628
12.284.2.40 greater()	628
12.284.2.41 greater_eq()	628
12.284.2.42 vand()	628
12.284.2.43 vor()	628
12.284.2.44 vxor()	628
12.284.2.45 vandnot()	628
12.284.2.46 floor()	628
12.284.2.47 ceil()	628
12.284.2.48 round()	628
12.284.2.49 hadd()	629
12.284.2.50 hadd_to_scal()	629
12.284.2.51 mod()	629
12.284.3 Field Documentation	629

12.284.3.1 vect_size	629
12.284.3.2 alignment	629
12.285 Simd256_impl< true, true, false, 2 > Struct Reference	629
12.285.1 Member Typedef Documentation	631
12.285.1.1 scalar_t	631
12.285.1.2 aligned_allocator	631
12.285.1.3 aligned_vector	631
12.285.1.4 is_same_element	631
12.285.1.5 simdHalf	632
12.285.1.6 vect_t	632
12.285.1.7 half_t	632
12.285.2 Member Function Documentation	632
12.285.2.1 type_string()	632
12.285.2.2 set1() [1/2]	632
12.285.2.3 set() [1/2]	632
12.285.2.4 gather() [1/2]	632
12.285.2.5 load() [1/2]	632
12.285.2.6 loadu() [1/2]	632
12.285.2.7 store() [1/2]	633
12.285.2.8 storeu() [1/2]	633
12.285.2.9 stream() [1/2]	633
12.285.2.10 sra()	633
12.285.2.11 greater()	633
12.285.2.12 lesser()	633
12.285.2.13 greater_eq()	633
12.285.2.14 lesser_eq()	633
12.285.2.15 mulhi()	633
12.285.2.16 mulx()	633
12.285.2.17 fmaddx()	634
12.285.2.18 fmaddxin()	634
12.285.2.19 fnmaddx()	634
12.285.2.20 fnmaddxin()	634
12.285.2.21 fmsubx()	634
12.285.2.22 fmsubxin()	634
12.285.2.23 hadd_to_scal()	634
12.285.2.24 valid()	634
12.285.2.25 compliant()	634
12.285.2.26 set1() [2/2]	635
12.285.2.27 set() [2/2]	635
12.285.2.28 gather() [2/2]	635
12.285.2.29 load() [2/2]	635
12.285.2.30 loadu() [2/2]	635

12.285.2.31 store() [2/2]	635
12.285.2.32 storeu() [2/2]	635
12.285.2.33 stream() [2/2]	635
12.285.2.34 sll()	636
12.285.2.35 srl()	636
12.285.2.36 shuffle()	636
12.285.2.37 unpacklo_intrinsic()	636
12.285.2.38 unpackhi_intrinsic()	636
12.285.2.39 unpacklo()	636
12.285.2.40 unpackhi()	636
12.285.2.41 unpacklohi()	636
12.285.2.42 pack_even()	636
12.285.2.43 pack_odd()	636
12.285.2.44 pack()	637
12.285.2.45 transpose()	637
12.285.2.46 blend() [1/2]	637
12.285.2.47 blend() [2/2]	637
12.285.2.48 add()	637
12.285.2.49 addin()	637
12.285.2.50 sub()	637
12.285.2.51 subin()	638
12.285.2.52 mullo()	638
12.285.2.53 mul()	638
12.285.2.54 fmadd()	638
12.285.2.55 fmaddin()	638
12.285.2.56 fnmadd()	638
12.285.2.57 fnmaddin()	638
12.285.2.58 fmsub()	638
12.285.2.59 fmsubin()	638
12.285.2.60 eq()	639
12.285.2.61 round()	639
12.285.2.62 mod()	639
12.285.2.63 zero()	639
12.285.3 Field Documentation	639
12.285.3.1 vect_size	639
12.285.3.2 alignment	639
12.286 Simd256_impl< true, true, false, 4 > Struct Reference	639
12.286.1 Member Typedef Documentation	643
12.286.1.1 scalar_t [1/2]	643
12.286.1.2 aligned_allocator [1/2]	643
12.286.1.3 aligned_vector [1/2]	643
12.286.1.4 is_same_element [1/2]	643

12.286.1.5 simdHalf [1/2]	643
12.286.1.6 scalar_t [2/2]	643
12.286.1.7 aligned_allocator [2/2]	643
12.286.1.8 aligned_vector [2/2]	643
12.286.1.9 is_same_element [2/2]	643
12.286.1.10 simdHalf [2/2]	643
12.286.1.11 vect_t [1/2]	643
12.286.1.12 vect_t [2/2]	643
12.286.1.13 half_t [1/2]	643
12.286.1.14 half_t [2/2]	644
12.286.2 Member Function Documentation	644
12.286.2.1 type_string() [1/2]	644
12.286.2.2 set1() [1/3]	644
12.286.2.3 set() [1/4]	644
12.286.2.4 gather() [1/3]	644
12.286.2.5 load() [1/3]	644
12.286.2.6 loadu() [1/3]	644
12.286.2.7 store() [1/3]	644
12.286.2.8 storeu() [1/3]	644
12.286.2.9 stream() [1/3]	644
12.286.2.10 sra() [1/2]	645
12.286.2.11 greater() [1/2]	645
12.286.2.12 lesser() [1/2]	645
12.286.2.13 greater_eq() [1/2]	645
12.286.2.14 lesser_eq() [1/2]	645
12.286.2.15 mulhi() [1/2]	645
12.286.2.16 mulx() [1/2]	645
12.286.2.17 fmaddx() [1/2]	645
12.286.2.18 fmaddxin() [1/2]	645
12.286.2.19 fnmaddx() [1/2]	646
12.286.2.20 fnmaddxin() [1/2]	646
12.286.2.21 fmsubx() [1/2]	646
12.286.2.22 fmsubxin() [1/2]	646
12.286.2.23 hadd_to_scal() [1/2]	646
12.286.2.24 type_string() [2/2]	646
12.286.2.25 set1() [2/3]	646
12.286.2.26 set() [2/4]	646
12.286.2.27 gather() [2/3]	646
12.286.2.28 load() [2/3]	647
12.286.2.29 loadu() [2/3]	647
12.286.2.30 store() [2/3]	647
12.286.2.31 storeu() [2/3]	647

12.286.2.32 stream()	[2/3]	647
12.286.2.33 sra()	[2/2]	647
12.286.2.34 greater()	[2/2]	647
12.286.2.35 lesser()	[2/2]	647
12.286.2.36 greater_eq()	[2/2]	647
12.286.2.37 lesser_eq()	[2/2]	647
12.286.2.38 mulhi()	[2/2]	648
12.286.2.39 mulx()	[2/2]	648
12.286.2.40 fmaddx()	[2/2]	648
12.286.2.41 fmaddxin()	[2/2]	648
12.286.2.42 fnmaddx()	[2/2]	648
12.286.2.43 fnmaddxin()	[2/2]	648
12.286.2.44 fmsubx()	[2/2]	648
12.286.2.45 fmsubxin()	[2/2]	648
12.286.2.46 hadd_to_scal()	[2/2]	648
12.286.2.47 valid()	[1/2]	649
12.286.2.48 valid()	[2/2]	649
12.286.2.49 compliant()	[1/2]	649
12.286.2.50 compliant()	[2/2]	649
12.286.2.51 set1()	[3/3]	649
12.286.2.52 set()	[3/4]	649
12.286.2.53 set()	[4/4]	649
12.286.2.54 gather()	[3/3]	650
12.286.2.55 load()	[3/3]	650
12.286.2.56 loadu()	[3/3]	650
12.286.2.57 store()	[3/3]	650
12.286.2.58 storeu()	[3/3]	650
12.286.2.59 stream()	[3/3]	650
12.286.2.60 sll()	[1/2]	650
12.286.2.61 sll()	[2/2]	650
12.286.2.62 srl()	[1/2]	650
12.286.2.63 srl()	[2/2]	650
12.286.2.64 shuffle_twice()	[1/2]	651
12.286.2.65 shuffle_twice()	[2/2]	651
12.286.2.66 shuffle()	[1/2]	651
12.286.2.67 shuffle()	[2/2]	651
12.286.2.68 unpacklo_intrinsic()	[1/2]	651
12.286.2.69 unpacklo_intrinsic()	[2/2]	651
12.286.2.70 unpackhi_intrinsic()	[1/2]	651
12.286.2.71 unpackhi_intrinsic()	[2/2]	651
12.286.2.72 unpacklo()	[1/2]	651
12.286.2.73 unpacklo()	[2/2]	651

12.286.2.74 unpackhi() [1/2]	652
12.286.2.75 unpackhi() [2/2]	652
12.286.2.76 unpacklohi() [1/2]	652
12.286.2.77 unpacklohi() [2/2]	652
12.286.2.78 pack_even() [1/2]	652
12.286.2.79 pack_even() [2/2]	652
12.286.2.80 pack_odd() [1/2]	652
12.286.2.81 pack_odd() [2/2]	652
12.286.2.82 pack() [1/2]	652
12.286.2.83 pack() [2/2]	653
12.286.2.84 transpose() [1/2]	653
12.286.2.85 transpose() [2/2]	653
12.286.2.86 blend() [1/2]	653
12.286.2.87 blend() [2/2]	653
12.286.2.88 add() [1/2]	653
12.286.2.89 add() [2/2]	654
12.286.2.90 addin() [1/2]	654
12.286.2.91 addin() [2/2]	654
12.286.2.92 sub() [1/2]	654
12.286.2.93 sub() [2/2]	654
12.286.2.94 subin() [1/2]	654
12.286.2.95 subin() [2/2]	654
12.286.2.96 mullo() [1/2]	654
12.286.2.97 mullo() [2/2]	654
12.286.2.98 mul() [1/2]	654
12.286.2.99 mul() [2/2]	655
12.286.2.100 fmadd() [1/2]	655
12.286.2.101 fmadd() [2/2]	655
12.286.2.102 fmaddin() [1/2]	655
12.286.2.103 fmaddin() [2/2]	655
12.286.2.104 fnmadd() [1/2]	655
12.286.2.105 fnmadd() [2/2]	655
12.286.2.106 fnmaddin() [1/2]	655
12.286.2.107 fnmaddin() [2/2]	655
12.286.2.108 fmsub() [1/2]	656
12.286.2.109 fmsub() [2/2]	656
12.286.2.110 fmsubin() [1/2]	656
12.286.2.111 fmsubin() [2/2]	656
12.286.2.112 eq() [1/2]	656
12.286.2.113 eq() [2/2]	656
12.286.2.114 round() [1/2]	656
12.286.2.115 round() [2/2]	656

12.286.2.116 mod() [1/2]	656
12.286.2.117 mod() [2/2]	657
12.286.2.118 zero() [1/2]	657
12.286.2.119 zero() [2/2]	657
12.286.2.120 vor()	657
12.286.2.121 vxor()	657
12.286.2.122 vand()	657
12.286.2.123 vandnot()	657
12.286.3 Field Documentation	657
12.286.3.1 vect_size	657
12.286.3.2 alignment	657
12.287 Simd256_impl< true, true, false, 8 > Struct Reference	658
12.287.1 Member Typedef Documentation	660
12.287.1.1 scalar_t	660
12.287.1.2 aligned_allocator	660
12.287.1.3 aligned_vector	660
12.287.1.4 is_same_element	660
12.287.1.5 simdHalf	660
12.287.1.6 vect_t	660
12.287.1.7 half_t	660
12.287.2 Member Function Documentation	660
12.287.2.1 type_string()	660
12.287.2.2 set1() [1/2]	660
12.287.2.3 set() [1/2]	660
12.287.2.4 gather() [1/2]	661
12.287.2.5 load() [1/2]	661
12.287.2.6 loadu() [1/2]	661
12.287.2.7 store() [1/2]	661
12.287.2.8 storeu() [1/2]	661
12.287.2.9 stream() [1/2]	661
12.287.2.10 sra()	661
12.287.2.11 greater()	661
12.287.2.12 lesser()	661
12.287.2.13 greater_eq()	661
12.287.2.14 lesser_eq()	662
12.287.2.15 mullo()	662
12.287.2.16 mulhi()	662
12.287.2.17 mulx()	662
12.287.2.18 fmaddx()	662
12.287.2.19 fmaddxin()	662
12.287.2.20 fnmaddx()	662
12.287.2.21 fnmaddxin()	662

12.287.2.22 fmsubx()	662
12.287.2.23 fmsubxin()	663
12.287.2.24 hadd_to_scal()	663
12.287.2.25 valid()	663
12.287.2.26 compliant()	663
12.287.2.27 set1() [2/2]	663
12.287.2.28 set() [2/2]	663
12.287.2.29 gather() [2/2]	663
12.287.2.30 get()	663
12.287.2.31 load() [2/2]	663
12.287.2.32 loadu() [2/2]	663
12.287.2.33 store() [2/2]	664
12.287.2.34 storeu() [2/2]	664
12.287.2.35 stream() [2/2]	664
12.287.2.36 sll()	664
12.287.2.37 srl()	664
12.287.2.38 shuffle()	664
12.287.2.39 unpacklo_intrinsic()	664
12.287.2.40 unpackhi_intrinsic()	664
12.287.2.41 unpacklo()	664
12.287.2.42 unpackhi()	664
12.287.2.43 unpacklohi()	665
12.287.2.44 pack_even()	665
12.287.2.45 pack_odd()	665
12.287.2.46 pack()	665
12.287.2.47 transpose()	665
12.287.2.48 blend()	665
12.287.2.49 add()	665
12.287.2.50 addin()	665
12.287.2.51 sub()	665
12.287.2.52 subin()	666
12.287.2.53 mul()	666
12.287.2.54 fmadd()	666
12.287.2.55 fmaddin()	666
12.287.2.56 fnmadd()	666
12.287.2.57 fnmaddin()	666
12.287.2.58 fmsub()	666
12.287.2.59 fmsubin()	666
12.287.2.60 eq()	666
12.287.2.61 round()	667
12.287.2.62 mask_high()	667
12.287.2.63 mulhi_fast()	667

12.287.2.64 mod()	667
12.287.2.65 signbits()	667
12.287.2.66 zero()	667
12.287.3 Field Documentation	667
12.287.3.1 vect_size	667
12.287.3.2 alignment	667
12.288 Simd256_impl< true, true, true, 2 > Struct Reference	667
12.288.1 Member Typedef Documentation	669
12.288.1.1 vect_t	669
12.288.1.2 half_t	669
12.288.1.3 scalar_t	669
12.288.1.4 simdHalf	670
12.288.1.5 aligned_allocator	670
12.288.1.6 aligned_vector	670
12.288.1.7 is_same_element	670
12.288.2 Member Function Documentation	670
12.288.2.1 type_string()	670
12.288.2.2 valid()	670
12.288.2.3 compliant()	670
12.288.2.4 set1()	670
12.288.2.5 set()	670
12.288.2.6 gather()	671
12.288.2.7 load()	671
12.288.2.8 loadu()	671
12.288.2.9 store()	671
12.288.2.10 storeu()	671
12.288.2.11 stream()	671
12.288.2.12 sll()	671
12.288.2.13 srl()	671
12.288.2.14 sra()	671
12.288.2.15 shuffle()	671
12.288.2.16 unpacklo_intrinsic()	672
12.288.2.17 unpackhi_intrinsic()	672
12.288.2.18 unpacklo()	672
12.288.2.19 unpackhi()	672
12.288.2.20 unpacklohi()	672
12.288.2.21 pack_even()	672
12.288.2.22 pack_odd()	672
12.288.2.23 pack()	672
12.288.2.24 transpose()	672
12.288.2.25 blend() [1/2]	673
12.288.2.26 blend() [2/2]	673

12.288.2.27	add()	673
12.288.2.28	addin()	673
12.288.2.29	sub()	673
12.288.2.30	subin()	673
12.288.2.31	mullo()	673
12.288.2.32	mul()	674
12.288.2.33	mulhi()	674
12.288.2.34	mulx()	674
12.288.2.35	fmadd()	674
12.288.2.36	fmaddin()	674
12.288.2.37	fmaddx()	674
12.288.2.38	fmaddxin()	674
12.288.2.39	fnmadd()	674
12.288.2.40	fnmaddin()	674
12.288.2.41	fnmaddx()	675
12.288.2.42	fnmaddxin()	675
12.288.2.43	fmsub()	675
12.288.2.44	fmsubin()	675
12.288.2.45	fmsubx()	675
12.288.2.46	fmsubxin()	675
12.288.2.47	eq()	675
12.288.2.48	greater()	675
12.288.2.49	lesser()	675
12.288.2.50	greater_eq()	676
12.288.2.51	lesser_eq()	676
12.288.2.52	hadd_to_scal()	676
12.288.2.53	round()	676
12.288.2.54	mod()	676
12.288.2.55	zero()	676
12.288.3	Field Documentation	676
12.288.3.1	vect_size	676
12.288.3.2	alignment	676
12.289	Simd256_impl< true, true, true, 4 > Struct Reference	676
12.289.1	Member Typedef Documentation	680
12.289.1.1	vect_t [1/2]	680
12.289.1.2	half_t [1/2]	680
12.289.1.3	scalar_t [1/2]	680
12.289.1.4	simdHalf [1/2]	680
12.289.1.5	aligned_allocator [1/2]	680
12.289.1.6	aligned_vector [1/2]	680
12.289.1.7	is_same_element [1/2]	680
12.289.1.8	vect_t [2/2]	680

12.289.1.9 half_t [2/2]	680
12.289.1.10 scalar_t [2/2]	680
12.289.1.11 simdHalf [2/2]	680
12.289.1.12 aligned_allocator [2/2]	681
12.289.1.13 aligned_vector [2/2]	681
12.289.1.14 is_same_element [2/2]	681
12.289.2 Member Function Documentation	681
12.289.2.1 type_string() [1/2]	681
12.289.2.2 valid() [1/2]	681
12.289.2.3 compliant() [1/2]	681
12.289.2.4 set1() [1/2]	681
12.289.2.5 set() [1/2]	681
12.289.2.6 gather() [1/2]	681
12.289.2.7 load() [1/2]	681
12.289.2.8 loadu() [1/2]	682
12.289.2.9 store() [1/2]	682
12.289.2.10 storeu() [1/2]	682
12.289.2.11 stream() [1/2]	682
12.289.2.12 sll() [1/2]	682
12.289.2.13 srl() [1/2]	682
12.289.2.14 sra() [1/2]	682
12.289.2.15 shuffle_twice() [1/2]	682
12.289.2.16 shuffle() [1/2]	682
12.289.2.17 unpacklo_intrinsic() [1/2]	682
12.289.2.18 unpackhi_intrinsic() [1/2]	683
12.289.2.19 unpacklo() [1/2]	683
12.289.2.20 unpackhi() [1/2]	683
12.289.2.21 unpacklohi() [1/2]	683
12.289.2.22 pack_even() [1/2]	683
12.289.2.23 pack_odd() [1/2]	683
12.289.2.24 pack() [1/2]	683
12.289.2.25 transpose() [1/2]	683
12.289.2.26 blend() [1/2]	684
12.289.2.27 add() [1/2]	684
12.289.2.28 addin() [1/2]	684
12.289.2.29 sub() [1/2]	684
12.289.2.30 subin() [1/2]	684
12.289.2.31 mullo() [1/2]	684
12.289.2.32 mul() [1/2]	684
12.289.2.33 mulhi() [1/2]	684
12.289.2.34 mulx() [1/2]	684
12.289.2.35 fmadd() [1/2]	684

12.289.2.36 fmaddin() [1/2]	685
12.289.2.37 fmaddx() [1/2]	685
12.289.2.38 fmaddxin() [1/2]	685
12.289.2.39 fnmadd() [1/2]	685
12.289.2.40 fnmaddin() [1/2]	685
12.289.2.41 fnmaddx() [1/2]	685
12.289.2.42 fnmaddxin() [1/2]	685
12.289.2.43 fmsub() [1/2]	685
12.289.2.44 fmsubin() [1/2]	685
12.289.2.45 fmsubx() [1/2]	686
12.289.2.46 fmsubxin() [1/2]	686
12.289.2.47 eq() [1/2]	686
12.289.2.48 greater() [1/2]	686
12.289.2.49 lesser() [1/2]	686
12.289.2.50 greater_eq() [1/2]	686
12.289.2.51 lesser_eq() [1/2]	686
12.289.2.52 hadd_to_scal() [1/2]	686
12.289.2.53 round() [1/2]	686
12.289.2.54 mod() [1/2]	687
12.289.2.55 type_string() [2/2]	687
12.289.2.56 valid() [2/2]	687
12.289.2.57 compliant() [2/2]	687
12.289.2.58 set1() [2/2]	687
12.289.2.59 set() [2/2]	687
12.289.2.60 gather() [2/2]	687
12.289.2.61 load() [2/2]	688
12.289.2.62 loadu() [2/2]	688
12.289.2.63 store() [2/2]	688
12.289.2.64 storeu() [2/2]	688
12.289.2.65 stream() [2/2]	688
12.289.2.66 sll() [2/2]	688
12.289.2.67 srl() [2/2]	688
12.289.2.68 sra() [2/2]	688
12.289.2.69 shuffle_twice() [2/2]	688
12.289.2.70 shuffle() [2/2]	688
12.289.2.71 unpacklo_intrinsic() [2/2]	689
12.289.2.72 unpackhi_intrinsic() [2/2]	689
12.289.2.73 unpacklo() [2/2]	689
12.289.2.74 unpackhi() [2/2]	689
12.289.2.75 unpacklohi() [2/2]	689
12.289.2.76 pack_even() [2/2]	689
12.289.2.77 pack_odd() [2/2]	689

12.289.2.78 pack() [2/2]	689
12.289.2.79 transpose() [2/2]	689
12.289.2.80 blend() [2/2]	690
12.289.2.81 add() [2/2]	690
12.289.2.82 addin() [2/2]	690
12.289.2.83 sub() [2/2]	690
12.289.2.84 subin() [2/2]	690
12.289.2.85 mullo() [2/2]	690
12.289.2.86 mul() [2/2]	690
12.289.2.87 mulhi() [2/2]	690
12.289.2.88 mulx() [2/2]	691
12.289.2.89 fmadd() [2/2]	691
12.289.2.90 fmaddin() [2/2]	691
12.289.2.91 fmaddx() [2/2]	691
12.289.2.92 fmaddxin() [2/2]	691
12.289.2.93 fnmadd() [2/2]	691
12.289.2.94 fnmaddin() [2/2]	691
12.289.2.95 fnmaddx() [2/2]	691
12.289.2.96 fnmaddxin() [2/2]	691
12.289.2.97 fmsub() [2/2]	692
12.289.2.98 fmsubin() [2/2]	692
12.289.2.99 fmsubx() [2/2]	692
12.289.2.100 fmsubxin() [2/2]	692
12.289.2.101 eq() [2/2]	692
12.289.2.102 greater() [2/2]	692
12.289.2.103 lesser() [2/2]	692
12.289.2.104 greater_eq() [2/2]	692
12.289.2.105 lesser_eq() [2/2]	692
12.289.2.106 hadd_to_scal() [2/2]	693
12.289.2.107 round() [2/2]	693
12.289.2.108 mod() [2/2]	693
12.289.2.109 zero() [1/2]	693
12.289.2.110 zero() [2/2]	693
12.289.2.111 vor()	693
12.289.2.112 vxor()	693
12.289.2.113 vand()	693
12.289.2.114 vandnot()	693
12.289.3 Field Documentation	693
12.289.3.1 vect_size	693
12.289.3.2 alignment	694
12.290 Simd256_impl< true, true, true, 8 > Struct Reference	694
12.290.1 Member Typedef Documentation	696

12.290.1.1 vect_t	696
12.290.1.2 half_t	696
12.290.1.3 scalar_t	696
12.290.1.4 simdHalf	696
12.290.1.5 aligned_allocator	696
12.290.1.6 aligned_vector	696
12.290.1.7 is_same_element	696
12.290.2 Member Function Documentation	696
12.290.2.1 type_string()	696
12.290.2.2 valid()	696
12.290.2.3 compliant()	696
12.290.2.4 set1()	696
12.290.2.5 set()	697
12.290.2.6 gather()	697
12.290.2.7 get()	697
12.290.2.8 load()	697
12.290.2.9 loadu()	697
12.290.2.10 store()	697
12.290.2.11 storeu()	697
12.290.2.12 stream()	697
12.290.2.13 sll()	697
12.290.2.14 srl()	697
12.290.2.15 sra()	698
12.290.2.16 shuffle()	698
12.290.2.17 unpacklo_intrinsic()	698
12.290.2.18 unpackhi_intrinsic()	698
12.290.2.19 unpacklo()	698
12.290.2.20 unpackhi()	698
12.290.2.21 unpacklohi()	698
12.290.2.22 pack_even()	698
12.290.2.23 pack_odd()	698
12.290.2.24 pack()	699
12.290.2.25 transpose()	699
12.290.2.26 blend()	699
12.290.2.27 add()	699
12.290.2.28 addin()	699
12.290.2.29 sub()	699
12.290.2.30 subin()	699
12.290.2.31 mullo()	699
12.290.2.32 mul()	699
12.290.2.33 mulhi()	700
12.290.2.34 mulx()	700

12.290.2.35 fmadd()	700
12.290.2.36 fmaddin()	700
12.290.2.37 fmaddx()	700
12.290.2.38 fmaddxin()	700
12.290.2.39 fnmadd()	700
12.290.2.40 fnmaddin()	700
12.290.2.41 fnmaddx()	700
12.290.2.42 fnmaddxin()	701
12.290.2.43 fmsub()	701
12.290.2.44 fmsubin()	701
12.290.2.45 fmsubx()	701
12.290.2.46 fmsubxin()	701
12.290.2.47 eq()	701
12.290.2.48 greater()	701
12.290.2.49 lesser()	701
12.290.2.50 greater_eq()	701
12.290.2.51 lesser_eq()	702
12.290.2.52 hadd_to_scal()	702
12.290.2.53 round()	702
12.290.2.54 mask_high()	702
12.290.2.55 mulhi_fast()	702
12.290.2.56 mod()	702
12.290.2.57 signbits()	702
12.290.2.58 zero()	702
12.290.3 Field Documentation	702
12.290.3.1 vect_size	702
12.290.3.2 alignment	702
12.291 Simd256fp_base Struct Reference	703
12.292 Simd256i_base Struct Reference	703
12.292.1 Member Typedef Documentation	703
12.292.1.1 vect_t	703
12.292.2 Member Function Documentation	703
12.292.2.1 zero()	703
12.293 Simd512_impl< ArithType, Int, Signed, Size > Struct Template Reference	703
12.294 Simd512_impl< true, false, true, 4 > Struct Reference	703
12.295 Simd512_impl< true, false, true, 8 > Struct Reference	704
12.295.1 Member Typedef Documentation	705
12.295.1.1 vect_t	705
12.295.1.2 scalar_t	705
12.295.1.3 aligned_allocator	705
12.295.1.4 aligned_vector	705
12.295.1.5 is_same_element	705

12.295.2 Member Function Documentation	705
12.295.2.1 type_string()	705
12.295.2.2 valid()	705
12.295.2.3 compliant()	705
12.295.2.4 zero()	706
12.295.2.5 set1()	706
12.295.2.6 set()	706
12.295.2.7 gather()	706
12.295.2.8 load()	706
12.295.2.9 loadu()	706
12.295.2.10 store()	706
12.295.2.11 storeu()	706
12.295.2.12 stream()	706
12.295.2.13 shuffle()	706
12.295.2.14 unpacklo_intrinsic()	707
12.295.2.15 unpackhi_intrinsic()	707
12.295.2.16 unpacklo()	707
12.295.2.17 unpackhi()	707
12.295.2.18 unpacklohi()	707
12.295.2.19 pack_even()	707
12.295.2.20 pack_odd()	707
12.295.2.21 pack()	707
12.295.2.22 transpose()	707
12.295.2.23 blend()	708
12.295.2.24 blendv()	708
12.295.2.25 add()	708
12.295.2.26 addin()	708
12.295.2.27 sub()	708
12.295.2.28 subin()	708
12.295.2.29 mul()	708
12.295.2.30 mulin()	708
12.295.2.31 div()	708
12.295.2.32 fmadd()	709
12.295.2.33 fmaddin()	709
12.295.2.34 fnmadd()	709
12.295.2.35 fnmaddin()	709
12.295.2.36 fmsub()	709
12.295.2.37 fmsubin()	709
12.295.2.38 eq()	709
12.295.2.39 lesser()	709
12.295.2.40 lesser_eq()	709
12.295.2.41 greater()	710

12.295.2.42 greater_eq()	710
12.295.2.43 floor()	710
12.295.2.44 ceil()	710
12.295.2.45 round()	710
12.295.2.46 hadd()	710
12.295.2.47 hadd_to_scal()	710
12.295.3 Field Documentation	710
12.295.3.1 vect_size	710
12.295.3.2 alignment	710
12.296 Simd512_impl< true, true, false, 8 > Struct Reference	710
12.296.1 Member Typedef Documentation	713
12.296.1.1 scalar_t	713
12.296.1.2 aligned_allocator	713
12.296.1.3 aligned_vector	713
12.296.1.4 is_same_element	713
12.296.1.5 simdHalf	713
12.296.1.6 vect_t	713
12.296.1.7 half_t	713
12.296.2 Member Function Documentation	713
12.296.2.1 type_string()	713
12.296.2.2 set1() [1/2]	713
12.296.2.3 set() [1/3]	713
12.296.2.4 gather() [1/2]	714
12.296.2.5 load() [1/2]	714
12.296.2.6 loadu() [1/2]	714
12.296.2.7 store() [1/2]	714
12.296.2.8 maskstore() [1/2]	714
12.296.2.9 storeu() [1/2]	714
12.296.2.10 stream() [1/2]	714
12.296.2.11 sra()	714
12.296.2.12 greater()	714
12.296.2.13 lesser()	714
12.296.2.14 greater_eq()	715
12.296.2.15 lesser_eq()	715
12.296.2.16 mullo()	715
12.296.2.17 mulhi()	715
12.296.2.18 mulx()	715
12.296.2.19 fmaddx()	715
12.296.2.20 fmaddxin()	715
12.296.2.21 fnmaddx()	715
12.296.2.22 fnmaddxin()	715
12.296.2.23 fmsubx()	716

12.296.2.24 fmsubxin()	716
12.296.2.25 hadd_to_scal()	716
12.296.2.26 valid()	716
12.296.2.27 compliant()	716
12.296.2.28 set1() [2/2]	716
12.296.2.29 set() [2/3]	716
12.296.2.30 set() [3/3]	716
12.296.2.31 gather() [2/2]	716
12.296.2.32 load() [2/2]	717
12.296.2.33 loadu() [2/2]	717
12.296.2.34 store() [2/2]	717
12.296.2.35 maskstore() [2/2]	717
12.296.2.36 storeu() [2/2]	717
12.296.2.37 stream() [2/2]	717
12.296.2.38 sll()	717
12.296.2.39 srl()	717
12.296.2.40 shuffle()	717
12.296.2.41 unpacklo_intrinsic()	718
12.296.2.42 unpackhi_intrinsic()	718
12.296.2.43 unpacklo()	718
12.296.2.44 unpackhi()	718
12.296.2.45 unpacklohi()	718
12.296.2.46 pack_even()	718
12.296.2.47 pack_odd()	718
12.296.2.48 pack()	718
12.296.2.49 transpose()	718
12.296.2.50 blend()	719
12.296.2.51 add()	719
12.296.2.52 addin()	719
12.296.2.53 sub()	719
12.296.2.54 subin()	719
12.296.2.55 mul()	719
12.296.2.56 fmadd()	719
12.296.2.57 fmaddin()	719
12.296.2.58 fnmadd()	719
12.296.2.59 fnmaddin()	720
12.296.2.60 fmsub()	720
12.296.2.61 fmsubin()	720
12.296.2.62 eq()	720
12.296.2.63 round()	720
12.296.2.64 mask_high()	720
12.296.2.65 mulhi_fast()	720

12.296.2.66 mod()	720
12.296.2.67 signbits()	721
12.296.2.68 zero()	721
12.296.2.69 vor()	721
12.296.2.70 vxor()	721
12.296.2.71 vand()	721
12.296.2.72 vandnot()	721
12.296.3 Field Documentation	721
12.296.3.1 vect_size	721
12.296.3.2 alignment	721
12.297 Simd512_impl< true, true, true, 8 > Struct Reference	721
12.297.1 Member Typedef Documentation	723
12.297.1.1 vect_t	723
12.297.1.2 half_t	723
12.297.1.3 scalar_t	723
12.297.1.4 simdHalf	724
12.297.1.5 aligned_allocator	724
12.297.1.6 aligned_vector	724
12.297.1.7 is_same_element	724
12.297.2 Member Function Documentation	724
12.297.2.1 type_string()	724
12.297.2.2 valid()	724
12.297.2.3 compliant()	724
12.297.2.4 set1()	724
12.297.2.5 set() [1/2]	724
12.297.2.6 set() [2/2]	724
12.297.2.7 gather()	725
12.297.2.8 load()	725
12.297.2.9 loadu()	725
12.297.2.10 store()	725
12.297.2.11 maskstore()	725
12.297.2.12 storeu()	725
12.297.2.13 stream()	725
12.297.2.14 sll()	725
12.297.2.15 srl()	725
12.297.2.16 sra()	725
12.297.2.17 shuffle()	726
12.297.2.18 unpacklo_intrinsic()	726
12.297.2.19 unpackhi_intrinsic()	726
12.297.2.20 unpacklo()	726
12.297.2.21 unpackhi()	726
12.297.2.22 unpacklohi()	726

12.297.2.23 pack_even()	726
12.297.2.24 pack_odd()	726
12.297.2.25 pack()	726
12.297.2.26 transpose()	727
12.297.2.27 blend()	727
12.297.2.28 add()	727
12.297.2.29 addin()	727
12.297.2.30 sub()	727
12.297.2.31 subin()	727
12.297.2.32 mullo()	727
12.297.2.33 mul()	727
12.297.2.34 mulhi()	727
12.297.2.35 mulx()	728
12.297.2.36 fmadd()	728
12.297.2.37 fmaddin()	728
12.297.2.38 fmaddx()	728
12.297.2.39 fmaddxin()	728
12.297.2.40 fnmadd()	728
12.297.2.41 fnmaddin()	728
12.297.2.42 fnmaddx()	728
12.297.2.43 fnmaddxin()	728
12.297.2.44 fmsub()	729
12.297.2.45 fmsubin()	729
12.297.2.46 fmsubx()	729
12.297.2.47 fmsubxin()	729
12.297.2.48 eq()	729
12.297.2.49 greater()	729
12.297.2.50 lesser()	729
12.297.2.51 greater_eq()	729
12.297.2.52 lesser_eq()	729
12.297.2.53 hadd_to_scal()	730
12.297.2.54 round()	730
12.297.2.55 mask_high()	730
12.297.2.56 mulhi_fast()	730
12.297.2.57 mod()	730
12.297.2.58 signbits()	730
12.297.2.59 zero()	730
12.297.2.60 vor()	730
12.297.2.61 vxor()	730
12.297.2.62 vand()	730
12.297.2.63 vandnot()	731
12.297.3 Field Documentation	731

12.297.3.1 vect_size	731
12.297.3.2 alignment	731
12.298 Simd512i_base Struct Reference	731
12.298.1 Member Typedef Documentation	731
12.298.1.1 vect_t	731
12.298.2 Member Function Documentation	731
12.298.2.1 zero()	731
12.298.2.2 vor()	731
12.298.2.3 vxor()	732
12.298.2.4 vand()	732
12.298.2.5 vandnot()	732
12.299 SimdChooser< T, bool, bool > Struct Template Reference	732
12.300 SimdChooser< T, false, b > Struct Template Reference	732
12.300.1 Member Typedef Documentation	732
12.300.1.1 value	732
12.301 SimdChooser< T, true, false > Struct Template Reference	732
12.301.1 Member Typedef Documentation	733
12.301.1.1 value	733
12.302 SimdChooser< T, true, true > Struct Template Reference	733
12.302.1 Member Typedef Documentation	733
12.302.1.1 value	733
12.303 simdToType< T > Struct Template Reference	733
12.304 Single Struct Reference	733
12.305 Sparse< Field, SparseMatrix_t, IdxT, PtrT > Struct Template Reference	733
12.306 Sparse< _Field, SparseMatrix_t::COO > Struct Template Reference	733
12.306.1 Member Typedef Documentation	734
12.306.1.1 Field	734
12.306.2 Field Documentation	734
12.306.2.1 col	734
12.306.2.2 row	734
12.306.2.3 dat	734
12.306.2.4 delayed	734
12.306.2.5 kmax	734
12.306.2.6 m	734
12.306.2.7 n	734
12.306.2.8 nnz	735
12.306.2.9 nElements	735
12.306.2.10 maxrow	735
12.307 Sparse< _Field, SparseMatrix_t::COO_ZO > Struct Template Reference	735
12.307.1 Member Typedef Documentation	735
12.307.1.1 Field	735
12.307.2 Field Documentation	736

12.307.2.1 cst	736
12.307.2.2 col	736
12.307.2.3 row	736
12.307.2.4 dat	736
12.307.2.5 delayed	736
12.307.2.6 kmax	736
12.307.2.7 m	736
12.307.2.8 n	736
12.307.2.9 nnz	736
12.307.2.10 nElements	736
12.307.2.11 maxrow	736
12.308 Sparse< _Field, SparseMatrix_t::CSR > Struct Template Reference	737
12.308.1 Member Typedef Documentation	737
12.308.1.1 Field	737
12.308.2 Field Documentation	737
12.308.2.1 delayed	737
12.308.2.2 kmax	737
12.308.2.3 m	737
12.308.2.4 n	737
12.308.2.5 nnz	738
12.308.2.6 nElements	738
12.308.2.7 maxrow	738
12.308.2.8 col	738
12.308.2.9 st	738
12.308.2.10 stend	738
12.308.2.11 dat	738
12.309 Sparse< _Field, SparseMatrix_t::CSR_HYB > Struct Template Reference	738
12.309.1 Member Typedef Documentation	739
12.309.1.1 Field	739
12.309.2 Field Documentation	739
12.309.2.1 delayed	739
12.309.2.2 col	739
12.309.2.3 st	739
12.309.2.4 dat	739
12.309.2.5 kmax	739
12.309.2.6 m	739
12.309.2.7 n	739
12.309.2.8 nnz	739
12.309.2.9 nElements	739
12.309.2.10 maxrow	740
12.309.2.11 nOnes	740
12.309.2.12 nMOnes	740

12.309.2.13 nOthers	740
12.310 Sparse< _Field, SparseMatrix_t::CSR_ZO > Struct Template Reference	740
12.310.1 Member Typedef Documentation	741
12.310.1.1 Field	741
12.310.2 Field Documentation	741
12.310.2.1 cst	741
12.310.2.2 delayed	741
12.310.2.3 kmax	741
12.310.2.4 m	741
12.310.2.5 n	741
12.310.2.6 nnz	741
12.310.2.7 nElements	741
12.310.2.8 maxrow	741
12.310.2.9 col	741
12.310.2.10 st	741
12.310.2.11 stend	742
12.310.2.12 dat	742
12.311 Sparse< _Field, SparseMatrix_t::ELL > Struct Template Reference	742
12.311.1 Member Typedef Documentation	742
12.311.1.1 Field	742
12.311.2 Field Documentation	742
12.311.2.1 delayed	742
12.311.2.2 kmax	743
12.311.2.3 m	743
12.311.2.4 n	743
12.311.2.5 ld	743
12.311.2.6 nnz	743
12.311.2.7 nElements	743
12.311.2.8 maxrow	743
12.311.2.9 col	743
12.311.2.10 dat	743
12.312 Sparse< _Field, SparseMatrix_t::ELL_simd > Struct Template Reference	743
12.312.1 Field Documentation	744
12.312.1.1 delayed	744
12.312.1.2 chunk	744
12.312.1.3 m	744
12.312.1.4 n	744
12.312.1.5 ld	744
12.312.1.6 kmax	744
12.312.1.7 nnz	744
12.312.1.8 nElements	744
12.312.1.9 maxrow	745

12.312.1.10 nChunks	745
12.312.1.11 col	745
12.312.1.12 dat	745
12.313 Sparse< _Field, SparseMatrix_t::ELL_simd_ZO > Struct Template Reference	745
12.313.1 Field Documentation	745
12.313.1.1 cst	745
12.313.1.2 delayed	746
12.313.1.3 chunk	746
12.313.1.4 m	746
12.313.1.5 n	746
12.313.1.6 ld	746
12.313.1.7 kmax	746
12.313.1.8 nnz	746
12.313.1.9 nElements	746
12.313.1.10 maxrow	746
12.313.1.11 nChunks	746
12.313.1.12 col	746
12.313.1.13 dat	746
12.314 Sparse< _Field, SparseMatrix_t::ELL_ZO > Struct Template Reference	747
12.314.1 Member Typedef Documentation	747
12.314.1.1 Field	747
12.314.2 Field Documentation	747
12.314.2.1 cst	747
12.314.2.2 delayed	747
12.314.2.3 kmax	747
12.314.2.4 m	747
12.314.2.5 n	748
12.314.2.6 ld	748
12.314.2.7 nnz	748
12.314.2.8 nElements	748
12.314.2.9 maxrow	748
12.314.2.10 col	748
12.314.2.11 dat	748
12.315 Sparse< _Field, SparseMatrix_t::HYB_ZO > Struct Template Reference	748
12.315.1 Member Typedef Documentation	749
12.315.1.1 Field	749
12.315.1.2 Self_t	749
12.315.2 Field Documentation	749
12.315.2.1 delayed	749
12.315.2.2 kmax	749
12.315.2.3 m	749
12.315.2.4 n	749

12.315.2.5 nnz	749
12.315.2.6 maxrow	749
12.315.2.7 nElements	749
12.315.2.8 dat	749
12.315.2.9 one	749
12.315.2.10 mone	750
12.316 Sparse< _Field, SparseMatrix_t::SELL > Struct Template Reference	750
12.316.1 Member Typedef Documentation	750
12.316.1.1 Field	750
12.316.2 Field Documentation	750
12.316.2.1 delayed	750
12.316.2.2 chunk	751
12.316.2.3 kmax	751
12.316.2.4 m	751
12.316.2.5 n	751
12.316.2.6 maxrow	751
12.316.2.7 sigma	751
12.316.2.8 nChunks	751
12.316.2.9 nnz	751
12.316.2.10 nElements	751
12.316.2.11 perm	751
12.316.2.12 st	751
12.316.2.13 chunkSize	751
12.316.2.14 col	752
12.316.2.15 dat	752
12.317 Sparse< _Field, SparseMatrix_t::SELL_ZO > Struct Template Reference	752
12.317.1 Member Typedef Documentation	752
12.317.1.1 Field	752
12.317.2 Field Documentation	753
12.317.2.1 cst	753
12.317.2.2 delayed	753
12.317.2.3 chunk	753
12.317.2.4 kmax	753
12.317.2.5 m	753
12.317.2.6 n	753
12.317.2.7 maxrow	753
12.317.2.8 sigma	753
12.317.2.9 nChunks	753
12.317.2.10 nnz	753
12.317.2.11 nElements	753
12.317.2.12 perm	753
12.317.2.13 st	754

12.317.2.14 chunkSize	754
12.317.2.15 col	754
12.317.2.16 dat	754
12.318 SpMat< Field, flag > Struct Template Reference	754
12.318.1 Field Documentation	754
12.318.1.1 _coo	754
12.318.1.2 _csr	754
12.318.1.3 _ell	754
12.319 StatsMatrix Struct Reference	754
12.319.1 Field Documentation	755
12.319.1.1 rowdim	755
12.319.1.2 coldim	755
12.319.1.3 nOnes	755
12.319.1.4 nMOnes	755
12.319.1.5 nOthers	755
12.319.1.6 nnz	755
12.319.1.7 maxRow	756
12.319.1.8 minRow	756
12.319.1.9 averageRow	756
12.319.1.10 deviationRow	756
12.319.1.11 maxCol	756
12.319.1.12 minCol	756
12.319.1.13 averageCol	756
12.319.1.14 deviationCol	756
12.319.1.15 minColDifference	756
12.319.1.16 maxColDifference	756
12.319.1.17 averageColDifference	756
12.319.1.18 deviationColDifference	756
12.319.1.19 minRowDifference	756
12.319.1.20 maxRowDifference	756
12.319.1.21 averageRowDifference	756
12.319.1.22 deviationRowDifference	757
12.319.1.23 nDenseRows	757
12.319.1.24 nDenseCols	757
12.319.1.25 nEmptyRows	757
12.319.1.26 nEmptyCols	757
12.319.1.27 nEmptyColsEnd	757
12.319.1.28 denseRows	757
12.319.1.29 denseCols	757
12.320 support_fast_mod< T > Struct Template Reference	757
12.321 support_fast_mod< double > Struct Reference	757
12.322 support_fast_mod< float > Struct Reference	758

12.323 support_fast_mod< int64_t > Struct Reference	758
12.324 support_simd< T > Struct Template Reference	758
12.325 support_simd_add< T > Struct Template Reference	759
12.326 support_simd_mod< T > Struct Template Reference	759
12.327 Test< Elt > Class Template Reference	759
12.327.1 Member Typedef Documentation	760
12.327.1.1 Field	760
12.327.1.2 Elt_ptr	760
12.327.1.3 Residu	760
12.327.1.4 enable_if_t	760
12.327.1.5 is_same_element	760
12.327.1.6 enable_if_no_simd_t	760
12.327.1.7 enable_if_simd128_t	760
12.327.1.8 enable_if_simd256_t	761
12.327.1.9 enable_if_simd512_t	761
12.327.2 Constructor & Destructor Documentation	761
12.327.2.1 Test()	761
12.327.3 Member Function Documentation	761
12.327.3.1 cardinality() [1/2]	761
12.327.3.2 cardinality() [2/2]	761
12.327.3.3 test_ftranspose()	761
12.327.3.4 doTests()	761
12.327.3.5 run()	761
12.327.4 Field Documentation	762
12.327.4.1 F	762
12.327.4.2 _mm	762
12.327.4.3 _nn	762
12.328 TestOneMethod< Simd > Class Template Reference	762
12.328.1 Member Typedef Documentation	763
12.328.1.1 Element	763
12.328.1.2 vect_t	763
12.328.1.3 vectElt	763
12.328.1.4 enable_if_t	763
12.328.2 Constructor & Destructor Documentation	763
12.328.2.1 TestOneMethod()	763
12.328.3 Member Function Documentation	763
12.328.3.1 evaluate_scalar_method() [1/3]	763
12.328.3.2 evaluate_scalar_method() [2/3]	764
12.328.3.3 evaluate_scalar_method() [3/3]	764
12.328.3.4 evaluate_simd_method() [1/2]	764
12.328.3.5 evaluate_simd_method() [2/2]	764
12.328.3.6 getStatus()	764

12.328.3.7	getTestName()	764
12.328.3.8	writeResultLine()	764
12.328.3.9	writeDebugData()	764
12.328.4	Field Documentation	764
12.328.4.1	vect_size	764
12.328.4.2	nb_lref	765
12.328.4.3	name	765
12.328.4.4	inputs	765
12.328.4.5	outputs_simd	765
12.328.4.6	outputs_scalar	765
12.329	tfn_minus Struct Reference	765
12.329.1	Member Function Documentation	765
12.329.1.1	operator>()	765
12.330	tfn_minus_eq Struct Reference	765
12.330.1	Member Function Documentation	766
12.330.1.1	operator>()	766
12.331	tfn_mul Struct Reference	766
12.331.1	Member Function Documentation	766
12.331.1.1	operator>()	766
12.332	tfn_mul_eq Struct Reference	766
12.332.1	Member Function Documentation	766
12.332.1.1	operator>()	766
12.333	tfn_plus Struct Reference	766
12.333.1	Member Function Documentation	767
12.333.1.1	operator>()	767
12.334	tfn_plus_eq Struct Reference	767
12.334.1	Member Function Documentation	767
12.334.1.1	operator>()	767
12.335	Threads Struct Reference	767
12.336	ThreeD Struct Reference	767
12.337	ThreeDAdaptive Struct Reference	767
12.338	ThreeDInPlace Struct Reference	768
12.339	TRSMHelper< ReclterTrait, ParSeqTrait > Struct Template Reference	768
12.339.1	Detailed Description	768
12.339.2	Constructor & Destructor Documentation	768
12.339.2.1	TRSMHelper() [1/3]	768
12.339.2.2	TRSMHelper() [2/3]	768
12.339.2.3	TRSMHelper() [3/3]	768
12.339.3	Member Function Documentation	769
12.339.3.1	pMMH() [1/2]	769
12.339.3.2	pMMH() [2/2]	769
12.339.4	Field Documentation	769

12.339.4.1 parseq	769
12.340 TwoD Struct Reference	769
12.341 TwoDAdaptive Struct Reference	769
12.342 UnparametricTag Struct Reference	769
12.342.1 Detailed Description	769
12.343 width< T > Struct Template Reference	770
12.343.1 Field Documentation	770
12.343.1.1 value	770
12.344 width< double > Struct Reference	770
12.344.1 Field Documentation	770
12.344.1.1 value	770
12.345 width< float > Struct Reference	770
12.345.1 Field Documentation	770
12.345.1.1 value	770
12.346 Winograd Struct Reference	770
12.347 WinogradPar Struct Reference	770
13 File Documentation	771
13.1 arithprog.C File Reference	771
13.1.1 Macro Definition Documentation	771
13.1.1.1 CUBE	771
13.1.1.2 GFOPS	771
13.1.2 Typedef Documentation	771
13.1.2.1 TTimer	771
13.1.3 Function Documentation	772
13.1.3.1 main()	772
13.2 fsyrk.C File Reference	772
13.2.1 Macro Definition Documentation	772
13.2.1.1 CUBE	772
13.2.1.2 GFOPS	772
13.2.2 Typedef Documentation	772
13.2.2.1 TTimer	772
13.2.3 Function Documentation	772
13.2.3.1 main()	772
13.3 fsytrf.C File Reference	773
13.3.1 Macro Definition Documentation	773
13.3.1.1 CUBE	773
13.3.1.2 GFOPS	773
13.3.2 Typedef Documentation	773
13.3.2.1 TTimer	773
13.3.3 Function Documentation	773
13.3.3.1 main()	773

13.4 ftrtri.C File Reference	773
13.4.1 Macro Definition Documentation	774
13.4.1.1 CUBE	774
13.4.1.2 GFOPS	774
13.4.2 Typedef Documentation	774
13.4.2.1 TTimer	774
13.4.3 Function Documentation	774
13.4.3.1 main()	774
13.5 winograd.C File Reference	774
13.5.1 Macro Definition Documentation	775
13.5.1.1 DOUBLE_TO_FLOAT_CROSSOVER	775
13.5.1.2 GFOPS	775
13.5.2 Typedef Documentation	775
13.5.2.1 TTimer	775
13.5.3 Function Documentation	775
13.5.3.1 balanced() [1/2]	775
13.5.3.2 balanced() [2/2]	775
13.5.3.3 main()	775
13.6 benchmark-charpoly-mp.C File Reference	775
13.6.1 Macro Definition Documentation	776
13.6.1.1 __FFLASFFPACK_FORCE_SEQ	776
13.6.2 Function Documentation	776
13.6.2.1 main()	776
13.7 benchmark-charpoly.C File Reference	776
13.7.1 Macro Definition Documentation	776
13.7.1.1 __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET	776
13.7.2 Function Documentation	776
13.7.2.1 run_with_field()	776
13.7.2.2 main()	777
13.8 benchmark-checkers.C File Reference	777
13.8.1 Macro Definition Documentation	777
13.8.1.1 ENABLE_ALL_CHECKINGS	777
13.8.1.2 _NR_TESTS	777
13.8.1.3 _MAX_SIZE_MATRICES	777
13.8.1.4 CUBE	777
13.8.2 Function Documentation	778
13.8.2.1 main()	778
13.9 benchmark-dgemm.C File Reference	778
13.9.1 Macro Definition Documentation	778
13.9.1.1 CBLAS_GEMM	778
13.9.2 Typedef Documentation	778
13.9.2.1 TTimer	778

13.9.2.2 Floats	778
13.9.3 Function Documentation	778
13.9.3.1 main()	778
13.10 benchmark-dgetrf.C File Reference	779
13.10.1 Macro Definition Documentation	779
13.10.1.1 __FFLASFFPACK_HAVE_DGETRF	779
13.10.2 Typedef Documentation	779
13.10.2.1 TTimer	779
13.10.3 Function Documentation	779
13.10.3.1 main()	779
13.11 benchmark-dgetri.C File Reference	779
13.11.1 Typedef Documentation	780
13.11.1.1 TTimer	780
13.11.2 Function Documentation	780
13.11.2.1 main()	780
13.12 benchmark-dsytrf.C File Reference	780
13.12.1 Macro Definition Documentation	780
13.12.1.1 EFFGFF	780
13.12.2 Typedef Documentation	780
13.12.2.1 TTimer	780
13.12.3 Function Documentation	781
13.12.3.1 main()	781
13.13 benchmark-dtrsm.C File Reference	781
13.13.1 Typedef Documentation	781
13.13.1.1 TTimer	781
13.13.2 Function Documentation	781
13.13.2.1 main()	781
13.14 benchmark-dtrtri.C File Reference	781
13.14.1 Macro Definition Documentation	782
13.14.1.1 __FFLASFFPACK_HAVE_DTRTRI	782
13.14.2 Typedef Documentation	782
13.14.2.1 TTimer	782
13.14.3 Function Documentation	782
13.14.3.1 main()	782
13.15 benchmark-fadd-lvl2.C File Reference	782
13.15.1 Macro Definition Documentation	782
13.15.1.1 __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET	782
13.15.2 Function Documentation	782
13.15.2.1 main()	782
13.16 benchmark-fdot.C File Reference	783
13.16.1 Macro Definition Documentation	783
13.16.1.1 __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET	783

13.16.2 Function Documentation	783
13.16.2.1 run_with_field()	783
13.16.2.2 main()	783
13.17 benchmark-fgemm-mp.C File Reference	783
13.17.1 Macro Definition Documentation	784
13.17.1.1 __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET	784
13.17.1.2 MG_DEFAULT	784
13.17.1.3 STD_RECINT_SIZE	784
13.17.2 Function Documentation	784
13.17.2.1 tmain()	784
13.17.2.2 main()	784
13.18 benchmark-fgemm-rns.C File Reference	784
13.18.1 Macro Definition Documentation	785
13.18.1.1 __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET	785
13.18.2 Typedef Documentation	785
13.18.2.1 RNS	785
13.18.2.2 Field	785
13.18.2.3 Element_ptr	785
13.18.2.4 ConstElement_ptr	785
13.18.2.5 THREADS	785
13.18.2.6 GRAIN	785
13.18.2.7 TWOD	785
13.18.2.8 TWODA	785
13.18.2.9 THREED	786
13.18.2.10 THREEDA	786
13.18.2.11 THREEDIP	786
13.18.2.12 PSeq	786
13.18.3 Function Documentation	786
13.18.3.1 main()	786
13.19 benchmark-fgemm.C File Reference	786
13.19.1 Macro Definition Documentation	786
13.19.1.1 CLASSIC_HYBRID	786
13.19.2 Function Documentation	786
13.19.2.1 main()	786
13.20 benchmark-fgemv-mp.C File Reference	787
13.20.1 Macro Definition Documentation	787
13.20.1.1 __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET	787
13.20.1.2 MG_DEFAULT	787
13.20.1.3 STD_RECINT_SIZE	787
13.20.2 Function Documentation	787
13.20.2.1 write_matrix()	787
13.20.2.2 tmain()	787

13.20.2.3 main()	788
13.21 benchmark-fgemv.C File Reference	788
13.21.1 Macro Definition Documentation	789
13.21.1.1 __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET	789
13.21.2 Function Documentation	789
13.21.2.1 fill_value()	789
13.21.2.2 genData()	789
13.21.2.3 check_result()	789
13.21.2.4 benchmark_with_timer()	789
13.21.2.5 benchmark_disp()	790
13.21.2.6 benchmark_in_Field()	790
13.21.2.7 benchmark_with_field() [1/2]	790
13.21.2.8 benchmark_with_field() [2/2]	791
13.21.2.9 main()	791
13.22 benchmark-fgesv.C File Reference	791
13.22.1 Macro Definition Documentation	791
13.22.1.1 __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET	791
13.22.2 Function Documentation	791
13.22.2.1 main()	791
13.23 benchmark-fsyr2k.C File Reference	791
13.23.1 Function Documentation	792
13.23.1.1 main()	792
13.24 benchmark-fsyrk.C File Reference	792
13.24.1 Macro Definition Documentation	792
13.24.1.1 __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET	792
13.24.2 Function Documentation	792
13.24.2.1 main()	792
13.25 benchmark-fsytrf.C File Reference	792
13.25.1 Macro Definition Documentation	793
13.25.1.1 __FFPACK_FSYTRF_BC_CROUT	793
13.25.1.2 __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET	793
13.25.1.3 CUBE	793
13.25.2 Function Documentation	793
13.25.2.1 main()	793
13.26 benchmark-ftsrm-mp.C File Reference	793
13.26.1 Macro Definition Documentation	794
13.26.1.1 __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET	794
13.26.2 Function Documentation	794
13.26.2.1 main()	794
13.27 benchmark-ftsrm.C File Reference	794
13.27.1 Macro Definition Documentation	794
13.27.1.1 __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET	794

13.27.2 Function Documentation	794
13.27.2.1 main()	794
13.28 benchmark-ftsv.C File Reference	794
13.28.1 Macro Definition Documentation	795
13.28.1.1 __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET	795
13.28.2 Function Documentation	795
13.28.2.1 main()	795
13.29 benchmark-fttri.C File Reference	795
13.29.1 Macro Definition Documentation	795
13.29.1.1 __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET	795
13.29.1.2 CUBE	795
13.29.2 Function Documentation	795
13.29.2.1 main()	795
13.30 benchmark-inverse.C File Reference	796
13.30.1 Macro Definition Documentation	796
13.30.1.1 CUBE	796
13.30.2 Function Documentation	796
13.30.2.1 main()	796
13.31 benchmark-lqup-mp.C File Reference	796
13.31.1 Function Documentation	796
13.31.1.1 main()	796
13.32 benchmark-lqup.C File Reference	797
13.32.1 Macro Definition Documentation	797
13.32.1.1 CUBE	797
13.32.2 Function Documentation	797
13.32.2.1 main()	797
13.33 benchmark-pluq.C File Reference	797
13.33.1 Macro Definition Documentation	798
13.33.1.1 __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET	798
13.33.1.2 CUBE	798
13.33.2 Typedef Documentation	798
13.33.2.1 Field	798
13.33.3 Function Documentation	798
13.33.3.1 verification_PLUQ()	798
13.33.3.2 Rec_Initialize()	798
13.33.3.3 main()	798
13.34 benchmark-quasisep.C File Reference	798
13.34.1 Macro Definition Documentation	799
13.34.1.1 __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET	799
13.34.2 Function Documentation	799
13.34.2.1 run_with_field()	799
13.34.2.2 main()	799

13.35 benchmark-storage-transpose.C File Reference	799
13.35.1 Function Documentation	800
13.35.1.1 main()	800
13.36 benchmark-wino.C File Reference	800
13.36.1 Macro Definition Documentation	800
13.36.1.1 CUBE	800
13.36.2 Function Documentation	800
13.36.2.1 launch_wino()	800
13.36.2.2 main()	801
13.37 config.h File Reference	801
13.37.1 Macro Definition Documentation	801
13.37.1.1 HAVE_BIG_ENDIAN	801
13.37.1.2 HAVE_BLAS	802
13.37.1.3 HAVE_CBLAS	802
13.37.1.4 HAVE_CXX11	802
13.37.1.5 HAVE_DLFCN_H	802
13.37.1.6 HAVE_FLOAT_H	802
13.37.1.7 HAVE_INT128	802
13.37.1.8 HAVE_INTPYPES_H	802
13.37.1.9 HAVE_LAPACK	802
13.37.1.10 HAVE_LIMITS_H	802
13.37.1.11 HAVE_PTHREAD_H	802
13.37.1.12 HAVE_STDDEF_H	802
13.37.1.13 HAVE_STDINT_H	802
13.37.1.14 HAVE_STDIO_H	802
13.37.1.15 HAVE_STDLIB_H	802
13.37.1.16 HAVE_STRINGS_H	802
13.37.1.17 HAVE_STRING_H	803
13.37.1.18 HAVE_SYS_STAT_H	803
13.37.1.19 HAVE_SYS_TIME_H	803
13.37.1.20 HAVE_SYS_TYPES_H	803
13.37.1.21 HAVE_UNISTD_H	803
13.37.1.22 LT_OBJDIR	803
13.37.1.23 OPENBLAS_NUM_THREADS	803
13.37.1.24 PACKAGE	803
13.37.1.25 PACKAGE_BUGREPORT	803
13.37.1.26 PACKAGE_NAME	803
13.37.1.27 PACKAGE_STRING	803
13.37.1.28 PACKAGE_TARNAME	803
13.37.1.29 PACKAGE_URL	803
13.37.1.30 PACKAGE_VERSION	803
13.37.1.31 SIZEOF_CHAR	803

13.37.1.32	SIZEOF_INT	804
13.37.1.33	SIZEOF_LONG	804
13.37.1.34	SIZEOF_LONG_LONG	804
13.37.1.35	SIZEOF_SHORT	804
13.37.1.36	SIZEOF__INT64_T	804
13.37.1.37	STDC_HEADERS	804
13.37.1.38	USE_OPENMP	804
13.37.1.39	VERSION	804
13.38	fflas-ffpack/config.h File Reference	804
13.38.1	Macro Definition Documentation	805
13.38.1.1	__FFLASFFPACK_HAVE_BIG_ENDIAN	805
13.38.1.2	__FFLASFFPACK_HAVE_BLAS	805
13.38.1.3	__FFLASFFPACK_HAVE_CBLAS	805
13.38.1.4	__FFLASFFPACK_HAVE_CXX11	805
13.38.1.5	__FFLASFFPACK_HAVE_DLFCN_H	805
13.38.1.6	__FFLASFFPACK_HAVE_FLOAT_H	805
13.38.1.7	__FFLASFFPACK_HAVE_INT128	805
13.38.1.8	__FFLASFFPACK_HAVE_INTPYPES_H	805
13.38.1.9	__FFLASFFPACK_HAVE_LAPACK	805
13.38.1.10	__FFLASFFPACK_HAVE_LIMITS_H	805
13.38.1.11	__FFLASFFPACK_HAVE_PTHREAD_H	806
13.38.1.12	__FFLASFFPACK_HAVE_STDDEF_H	806
13.38.1.13	__FFLASFFPACK_HAVE_STDINT_H	806
13.38.1.14	__FFLASFFPACK_HAVE_STDIO_H	806
13.38.1.15	__FFLASFFPACK_HAVE_STDLIB_H	806
13.38.1.16	__FFLASFFPACK_HAVE_STRINGS_H	806
13.38.1.17	__FFLASFFPACK_HAVE_STRING_H	806
13.38.1.18	__FFLASFFPACK_HAVE_SYS_STAT_H	806
13.38.1.19	__FFLASFFPACK_HAVE_SYS_TIME_H	806
13.38.1.20	__FFLASFFPACK_HAVE_SYS_TYPES_H	806
13.38.1.21	__FFLASFFPACK_HAVE_UNISTD_H	806
13.38.1.22	__FFLASFFPACK_LT_OBJDIR	806
13.38.1.23	__FFLASFFPACK_OPENBLAS_NUM_THREADS	806
13.38.1.24	__FFLASFFPACK_PACKAGE	806
13.38.1.25	__FFLASFFPACK_PACKAGE_BUGREPORT	806
13.38.1.26	__FFLASFFPACK_PACKAGE_NAME	807
13.38.1.27	__FFLASFFPACK_PACKAGE_STRING	807
13.38.1.28	__FFLASFFPACK_PACKAGE_TARNAME	807
13.38.1.29	__FFLASFFPACK_PACKAGE_URL	807
13.38.1.30	__FFLASFFPACK_PACKAGE_VERSION	807
13.38.1.31	__FFLASFFPACK_SIZEOF_CHAR	807
13.38.1.32	__FFLASFFPACK_SIZEOF_INT	807

13.38.1.33	<code>__FFLASFFPACK_SIZEOF_LONG</code>	807
13.38.1.34	<code>__FFLASFFPACK_SIZEOF_LONG_LONG</code>	807
13.38.1.35	<code>__FFLASFFPACK_SIZEOF_SHORT</code>	807
13.38.1.36	<code>__FFLASFFPACK_SIZEOF__INT64_T</code>	807
13.38.1.37	<code>__FFLASFFPACK_STDC_HEADERS</code>	807
13.38.1.38	<code>__FFLASFFPACK_USE_OPENMP</code>	807
13.38.1.39	<code>__FFLASFFPACK_VERSION</code>	807
13.39	mainpage.doxy File Reference	808
13.40	autotune/charpoly.C File Reference	808
13.40.1	Macro Definition Documentation	808
13.40.1.1	CUBE	808
13.40.1.2	GFOPS	808
13.40.2	Typedef Documentation	808
13.40.2.1	TTimer	808
13.40.3	Function Documentation	808
13.40.3.1	main()	808
13.41	examples/charpoly.C File Reference	808
13.41.1	Function Documentation	809
13.41.1.1	main()	809
13.42	det.C File Reference	809
13.42.1	Function Documentation	809
13.42.1.1	main()	809
13.43	matmul.C File Reference	809
13.43.1	Function Documentation	809
13.43.1.1	main()	809
13.44	autotune/pluq.C File Reference	810
13.44.1	Macro Definition Documentation	810
13.44.1.1	CUBE	810
13.44.1.2	GFOPS	810
13.44.2	Typedef Documentation	810
13.44.2.1	TTimer	810
13.44.3	Function Documentation	810
13.44.3.1	main()	810
13.45	examples/pluq.C File Reference	810
13.45.1	Function Documentation	811
13.45.1.1	main()	811
13.46	rank.C File Reference	811
13.46.1	Function Documentation	811
13.46.1.1	main()	811
13.47	solve.C File Reference	811
13.47.1	Function Documentation	811
13.47.1.1	main()	811

13.48 checker_charpoly.inl File Reference	812
13.48.1 Macro Definition Documentation	812
13.48.1.1 __FFLASFFPACK_checker_charpoly_INL	812
13.49 checker_det.inl File Reference	812
13.49.1 Macro Definition Documentation	812
13.49.1.1 __FFLASFFPACK_checker_det_INL	812
13.50 checker_empty.h File Reference	812
13.51 checker_fgemm.inl File Reference	813
13.51.1 Macro Definition Documentation	813
13.51.1.1 __FFLASFFPACK_checker_fgemm_INL	813
13.52 checker_ftsm.inl File Reference	813
13.52.1 Macro Definition Documentation	813
13.52.1.1 __FFLASFFPACK_checker_ftsm_INL	813
13.53 checker_invert.inl File Reference	813
13.53.1 Macro Definition Documentation	814
13.53.1.1 __FFLASFFPACK_checker_invert_INL	814
13.54 checker_pluq.inl File Reference	814
13.54.1 Macro Definition Documentation	814
13.54.1.1 __FFLASFFPACK_checker_pluq_INL	814
13.55 checkers.doxy File Reference	814
13.56 checkers_fflas.h File Reference	814
13.57 checkers_fflas.inl File Reference	815
13.57.1 Macro Definition Documentation	815
13.57.1.1 FFLASFFPACK_checkers_fflas_inl_H	815
13.58 checkers_ffpack.h File Reference	815
13.59 checkers_ffpack.inl File Reference	816
13.59.1 Macro Definition Documentation	816
13.59.1.1 FFLASFFPACK_checkers_ffpack_inl_H	816
13.60 config-blas.h File Reference	816
13.60.1 Macro Definition Documentation	817
13.60.1.1 CBLAS_INT	817
13.60.1.2 CBLAS_ENUM_DEFINED_H	817
13.60.1.3 CBLAS_EXTERNALS	817
13.60.1.4 blas_enum	817
13.60.2 Enumeration Type Documentation	817
13.60.2.1 CBLAS_ORDER	817
13.60.2.2 CBLAS_TRANSPOSE	818
13.60.2.3 CBLAS_UPLO	818
13.60.2.4 CBLAS_DIAG	818
13.60.2.5 CBLAS_SIDE	818
13.60.3 Function Documentation	818
13.60.3.1 daxpy_()	818

13.60.3.2 saxpy_()	818
13.60.3.3 ddot_()	819
13.60.3.4 sdot_()	819
13.60.3.5 dasum_()	819
13.60.3.6 idamax_()	819
13.60.3.7 dnrm2_()	819
13.60.3.8 dgemv_()	819
13.60.3.9 sgemv_()	820
13.60.3.10 dger_()	820
13.60.3.11 sger_()	820
13.60.3.12 dcopy_()	820
13.60.3.13 scopy_()	820
13.60.3.14 dscal_()	821
13.60.3.15 sscal_()	821
13.60.3.16 dtrsm_()	821
13.60.3.17 strsm_()	821
13.60.3.18 dtrmm_()	821
13.60.3.19 strmm_()	822
13.60.3.20 sgemm_()	822
13.60.3.21 dgemm_()	822
13.60.3.22 cblas_dsyrk()	822
13.61 fflas-ffpack-config.h File Reference	823
13.61.1 Detailed Description	823
13.61.2 Macro Definition Documentation	823
13.61.2.1 GCC_VERSION	823
13.62 fflas-ffpack-default-thresholds.h File Reference	823
13.62.1 Macro Definition Documentation	823
13.62.1.1 __FFLASFFPACK_WINOTHRESHOLD	823
13.62.1.2 __FFLASFFPACK_WINOTHRESHOLD_FLT	823
13.62.1.3 __FFLASFFPACK_WINOTHRESHOLD_BAL	824
13.62.1.4 __FFLASFFPACK_WINOTHRESHOLD_BAL_FLT	824
13.62.1.5 __FFLASFFPACK_PLUQ_THRESHOLD	824
13.62.1.6 __FFLASFFPACK_CHARPOLY_LUKrylov_ArithProg_THRESHOLD	824
13.62.1.7 __FFLASFFPACK_CHARPOLY_Danilevskii_LUKrylov_THRESHOLD	824
13.62.1.8 __FFLASFFPACK_ARITHPROG_THRESHOLD	824
13.62.1.9 __FFLASFFPACK_FTRTRI_THRESHOLD	824
13.62.1.10 __FFLASFFPACK_FSYTRF_THRESHOLD	824
13.62.1.11 __FFLASFFPACK_FSYRK_THRESHOLD	824
13.63 fflas-ffpack-thresholds.h File Reference	824
13.64 fflas-ffpack.doxy File Reference	824
13.65 fflas-ffpack.h File Reference	824
13.65.1 Detailed Description	824

13.66 fflas.doxy File Reference	824
13.67 fflas.h File Reference	824
13.67.1 Detailed Description	825
13.67.2 Macro Definition Documentation	826
13.67.2.1 WINOTHRESHOLD	826
13.67.2.2 DOUBLE_TO_FLOAT_CROSSOVER	826
13.68 fflas_bounds.inl File Reference	826
13.68.1 Macro Definition Documentation	827
13.68.1.1 __FFLASFFPACK_fflas_bounds_INL	827
13.68.1.2 FFLAS_INT_TYPE	827
13.69 fflas_enum.h File Reference	827
13.70 fflas_fadd.h File Reference	827
13.71 fflas_fadd.inl File Reference	829
13.71.1 Macro Definition Documentation	830
13.71.1.1 __FFLASFFPACK_fadd_INL	830
13.72 fflas_fassign.h File Reference	830
13.73 fflas_fassign.inl File Reference	830
13.73.1 Macro Definition Documentation	831
13.73.1.1 __FFLASFFPACK_fassign_INL	831
13.74 fflas_faxpy.inl File Reference	831
13.74.1 Macro Definition Documentation	832
13.74.1.1 __FFLASFFPACK_faxpy_INL	832
13.75 fflas_fdot.inl File Reference	832
13.75.1 Macro Definition Documentation	833
13.75.1.1 __FFLASFFPACK_fdot_INL	833
13.76 fflas_fgemm.inl File Reference	833
13.76.1 Macro Definition Documentation	835
13.76.1.1 __FFLASFFPACK_fgemm_INL	835
13.77 fgemm_classical.inl File Reference	835
13.78 fgemm_classical_mp.inl File Reference	835
13.78.1 Detailed Description	836
13.78.2 Macro Definition Documentation	837
13.78.2.1 __FFPACK_fgemm_classical_INL	837
13.79 fgemm_winograd.inl File Reference	837
13.79.1 Macro Definition Documentation	838
13.79.1.1 __FFLASFFPACK_fflas_fflas_fgemm_winograd_INL	838
13.79.1.2 NEWWINO	838
13.80 matmul.doxy File Reference	838
13.81 schedule_bini.inl File Reference	838
13.81.1 Detailed Description	838
13.81.2 Macro Definition Documentation	839
13.81.2.1 __FFLASFFPACK_fgemm_bini_INL	839

13.82	schedule_winograd.inl File Reference	839
13.82.1	Macro Definition Documentation	839
13.82.1.1	__FFLASFFPACK_fgemm_winograd_INL	839
13.83	schedule_winograd_acc.inl File Reference	839
13.83.1	Macro Definition Documentation	840
13.83.1.1	__FFLASFFPACK_fgemm_winograd_acc_INL	840
13.84	schedule_winograd_acc_ip.inl File Reference	840
13.84.1	Macro Definition Documentation	841
13.84.1.1	__FFLASFFPACK_fgemm_winograd_acc_ip_INL	841
13.85	schedule_winograd_ip.inl File Reference	841
13.85.1	Macro Definition Documentation	841
13.85.1.1	__FFLASFFPACK_fgemm_winograd_ip_INL	841
13.86	fflas_fgemv.inl File Reference	841
13.86.1	Macro Definition Documentation	843
13.86.1.1	__FFLASFFPACK_fgemv_INL	843
13.87	fflas_fgemv_mp.inl File Reference	843
13.87.1	Macro Definition Documentation	844
13.87.1.1	__FFLASFFPACK_fgemv_mp_INL	844
13.88	fflas_fger.inl File Reference	844
13.88.1	Macro Definition Documentation	845
13.88.1.1	__FFLASFFPACK_fger_INL	845
13.89	fflas_fger_mp.inl File Reference	845
13.89.1	Macro Definition Documentation	845
13.89.1.1	__FFPACK_fger_mp_INL	845
13.90	fflas_freduce.h File Reference	846
13.91	fflas_freduce.inl File Reference	847
13.91.1	Macro Definition Documentation	848
13.91.1.1	__FFLASFFPACK_fflas_freduce_INL	848
13.91.1.2	FFLASFFPACK_COPY_REDUCE	848
13.92	fflas_freduce_mp.inl File Reference	848
13.92.1	Macro Definition Documentation	848
13.92.1.1	__FFLASFFPACK_fflas_freduce_mp_INL	848
13.93	fflas_freivalds.inl File Reference	849
13.93.1	Macro Definition Documentation	849
13.93.1.1	__FFLASFFPACK_freivalds_INL	849
13.94	fflas_fscal.h File Reference	849
13.95	fflas_fscal.inl File Reference	849
13.95.1	Macro Definition Documentation	850
13.95.1.1	__FFLASFFPACK_fscal_INL	850
13.96	fflas_fscal_mp.inl File Reference	851
13.96.1	Macro Definition Documentation	851
13.96.1.1	__FFLASFFPACK_fscal_mp_INL	851

13.97 fflas_fsy2k.inl File Reference	851
13.97.1 Macro Definition Documentation	852
13.97.1.1 __FFLASFFPACK_fflas_fsy2k_INL	852
13.98 fflas_fsyrk.inl File Reference	852
13.98.1 Macro Definition Documentation	854
13.98.1.1 __FFLASFFPACK_fflas_fsyrk_INL	854
13.99 fflas_fsyrk_strassen.inl File Reference	854
13.99.1 Macro Definition Documentation	855
13.99.1.1 __FFLASFFPACK_fflas_fsyrk_strassen_INL	855
13.100 fflas_ftrmm.inl File Reference	855
13.100.1 Macro Definition Documentation	855
13.100.1.1 __FFLASFFPACK_ftrmm_INL	855
13.101 fflas_ftrsm.inl File Reference	855
13.101.1 Macro Definition Documentation	856
13.101.1.1 __FFLASFFPACK_ftrsm_INL	856
13.102 fflas_ftrsm_mp.inl File Reference	856
13.102.1 Detailed Description	856
13.102.2 Macro Definition Documentation	856
13.102.2.1 __FFPACK_ftrsm_mp_INL	856
13.103 fflas_ftrsv.inl File Reference	857
13.103.1 Macro Definition Documentation	857
13.103.1.1 __FFLASFFPACK_ftrsv_INL	857
13.104 fflas_helpers.inl File Reference	857
13.104.1 Macro Definition Documentation	858
13.104.1.1 __FFLASFFPACK_fflas_fflas_mmhelper_INL	858
13.105 igemm.doxy File Reference	858
13.106 igemm.h File Reference	858
13.107 igemm.inl File Reference	859
13.107.1 Macro Definition Documentation	859
13.107.1.1 __FFLASFFPACK_fflas_igemm_igemm_INL	859
13.108 igemm_kernels.h File Reference	859
13.109 igemm_kernels.inl File Reference	860
13.109.1 Macro Definition Documentation	861
13.109.1.1 __FFLASFFPACK_fflas_igemm_igemm_kernels_INL	861
13.110 igemm_tools.h File Reference	861
13.111 igemm_tools.inl File Reference	861
13.111.1 Macro Definition Documentation	861
13.111.1.1 __FFLASFFPACK_fflas_igemm_igemm_tools_INL	861
13.112 fflas_level1.inl File Reference	861
13.112.1 Macro Definition Documentation	864
13.112.1.1 __FFLASFFPACK_fflas_fflas_level1_INL	864
13.113 fflas_level2.inl File Reference	864

13.113.1 Macro Definition Documentation	866
13.113.1.1 __FFLASFFPACK_fflas_fflas_level2_INL	866
13.114 fflas_level3.inl File Reference	866
13.114.1 Macro Definition Documentation	869
13.114.1.1 __FFLASFFPACK_fflas_fflas_level3_INL	869
13.114.1.2 __FFLAS__TRSM_READONLY	869
13.115 fflas_pfgemm.inl File Reference	869
13.115.1 Macro Definition Documentation	869
13.115.1.1 __FFLASFFPACK_fflas_pfgemm_INL	869
13.115.1.2 __FFLASFFPACK_SEQPARTHRESHOLD	869
13.115.1.3 __FFLASFFPACK_DIMKPENALTY	870
13.116 fflas_pftrsm.inl File Reference	870
13.116.1 Macro Definition Documentation	870
13.116.1.1 __FFLASFFPACK_fflas_pftrsm_INL	870
13.116.1.2 PTRSM_HYBRID_THRESHOLD	870
13.117 fflas_simd.h File Reference	870
13.117.1 Macro Definition Documentation	871
13.117.1.1 SIMD_INT	871
13.117.1.2 INLINE	871
13.117.1.3 CONST	871
13.117.1.4 PURE	871
13.117.1.5 NORML_MOD	871
13.117.1.6 FLOAT_MOD	872
13.117.2 Typedef Documentation	872
13.117.2.1 Simd	872
13.118 simd.dox File Reference	872
13.119 simd128.inl File Reference	872
13.119.1 Macro Definition Documentation	872
13.119.1.1 __FFLASFFPACK_fflas_ffpack_utils_simd128_INL	872
13.119.2 Typedef Documentation	873
13.119.2.1 Simd128	873
13.120 simd128_double.inl File Reference	873
13.120.1 Macro Definition Documentation	873
13.120.1.1 __FFLASFFPACK_fflas_ffpack_utils_simd128_double_INL	873
13.121 simd128_float.inl File Reference	873
13.121.1 Macro Definition Documentation	873
13.121.1.1 __FFLASFFPACK_fflas_ffpack_utils_simd128_float_INL	873
13.122 simd128_int16.inl File Reference	873
13.122.1 Macro Definition Documentation	874
13.122.1.1 __FFLASFFPACK_fflas_ffpack_utils_simd128_int16_INL	874
13.123 simd128_int32.inl File Reference	874
13.123.1 Macro Definition Documentation	874

13.123.1.1 __FFLASFFPACK_fflas_ffpack_utils_simd128_int32_INL	874
13.124 simd128_int64.inl File Reference	874
13.124.1 Macro Definition Documentation	875
13.124.1.1 __FFLASFFPACK_fflas_ffpack_utils_simd128_int64_INL	875
13.124.1.2 vect_t	875
13.125 simd256.inl File Reference	875
13.125.1 Macro Definition Documentation	875
13.125.1.1 __FFLASFFPACK_fflas_ffpack_utils_simd256_INL	875
13.125.2 Typedef Documentation	875
13.125.2.1 Simd256	875
13.126 simd256_double.inl File Reference	875
13.126.1 Macro Definition Documentation	876
13.126.1.1 __FFLASFFPACK_fflas_ffpack_utils_simd256_double_INL	876
13.127 simd256_float.inl File Reference	876
13.127.1 Macro Definition Documentation	876
13.127.1.1 __FFLASFFPACK_fflas_ffpack_utils_simd256_float_INL	876
13.128 simd256_int16.inl File Reference	876
13.128.1 Macro Definition Documentation	877
13.128.1.1 __FFLASFFPACK_fflas_ffpack_utils_simd256_int16_INL	877
13.129 simd256_int32.inl File Reference	877
13.129.1 Macro Definition Documentation	877
13.129.1.1 __FFLASFFPACK_fflas_ffpack_utils_simd256_int32_INL	877
13.130 simd256_int64.inl File Reference	877
13.130.1 Macro Definition Documentation	877
13.130.1.1 __FFLASFFPACK_fflas_ffpack_utils_simd256_int64_INL	877
13.130.1.2 vect_t	878
13.131 simd512.inl File Reference	878
13.131.1 Macro Definition Documentation	878
13.131.1.1 __FFLASFFPACK_simd512_INL	878
13.131.2 Typedef Documentation	878
13.131.2.1 Simd512	878
13.132 simd512_double.inl File Reference	878
13.132.1 Macro Definition Documentation	878
13.132.1.1 __FFLASFFPACK_simd512_double_INL	878
13.133 simd512_float.inl File Reference	879
13.133.1 Macro Definition Documentation	879
13.133.1.1 __FFLASFFPACK_simd512_float_INL	879
13.134 simd512_int32.inl File Reference	879
13.134.1 Macro Definition Documentation	879
13.134.1.1 __FFLASFFPACK_simd512_int32_INL	879
13.135 simd512_int64.inl File Reference	879
13.135.1 Macro Definition Documentation	880

13.135.1.1 _simd512_int64_INL	880
13.135.1.2 vect_t	880
13.136 simd_modular.inl File Reference	880
13.137 fflas_sparse.h File Reference	880
13.137.1 Macro Definition Documentation	884
13.137.1.1 index_t	884
13.137.1.2 ROUND_DOWN	884
13.137.1.3 __FFLASFFPACK_CACHE_LINE_SIZE	884
13.137.1.4 assume_aligned	884
13.137.1.5 DENSE_THRESHOLD	884
13.138 fflas_sparse.inl File Reference	884
13.138.1 Macro Definition Documentation	886
13.138.1.1 __FFLASFFPACK_fflas_fflas_sparse_INL	886
13.139 coo.h File Reference	886
13.140 coo_spm.inl File Reference	887
13.140.1 Macro Definition Documentation	888
13.140.1.1 __FFLASFFPACK_fflas_sparse_coo_spm_INL	888
13.141 coo_spmv.inl File Reference	888
13.141.1 Macro Definition Documentation	889
13.141.1.1 __FFLASFFPACK_fflas_sparse_coo_spmv_INL	889
13.142 coo_utils.inl File Reference	889
13.142.1 Macro Definition Documentation	889
13.142.1.1 __FFLASFFPACK_fflas_sparse_coo_utils_INL	889
13.143 csr.h File Reference	889
13.144 csr_pspmm.inl File Reference	890
13.144.1 Macro Definition Documentation	890
13.144.1.1 __FFLASFFPACK_fflas_sparse_CSR_pspmm_INL	890
13.145 csr_pspmv.inl File Reference	891
13.145.1 Macro Definition Documentation	891
13.145.1.1 __FFLASFFPACK_fflas_sparse_CSR_pspmv_INL	891
13.146 csr_spm.inl File Reference	891
13.146.1 Macro Definition Documentation	892
13.146.1.1 __FFLASFFPACK_fflas_sparse_CSR_spm_INL	892
13.147 csr_spmv.inl File Reference	892
13.147.1 Macro Definition Documentation	893
13.147.1.1 __FFLASFFPACK_fflas_sparse_CSR_spmv_INL	893
13.148 csr_utils.inl File Reference	893
13.149 csr_hyb.h File Reference	894
13.150 csr_hyb_pspmm.inl File Reference	894
13.150.1 Macro Definition Documentation	895
13.150.1.1 __FFLASFFPACK_fflas_sparse_CSR_HYB_pspmm_INL	895
13.151 csr_hyb_spmv.inl File Reference	895

13.151.1 Macro Definition Documentation	895
13.151.1.1 __FFLASFFPACK_fflas_sparse_CSR_HYB_pspmv_INL	895
13.152 csr_hyb_spm্ম.inl File Reference	895
13.152.1 Macro Definition Documentation	896
13.152.1.1 __FFLASFFPACK_fflas_sparse_CSR_HYB_spm্ম_INL	896
13.153 csr_hyb_spmmv.inl File Reference	896
13.153.1 Macro Definition Documentation	896
13.153.1.1 __FFLASFFPACK_fflas_sparse_CSR_HYB_spmmv_INL	896
13.154 csr_hyb_utils.inl File Reference	896
13.154.1 Macro Definition Documentation	897
13.154.1.1 __FFLASFFPACK_fflas_sparse_CSR_HYB_utils_INL	897
13.155 ell.h File Reference	897
13.156 ell_pspmm.inl File Reference	897
13.156.1 Macro Definition Documentation	898
13.156.1.1 __FFLASFFPACK_fflas_sparse_ELL_pspmm_INL	898
13.157 ell_pspmv.inl File Reference	898
13.157.1 Macro Definition Documentation	899
13.157.1.1 __FFLASFFPACK_fflas_sparse_ELL_pspmv_INL	899
13.158 ell_spm্ম.inl File Reference	899
13.158.1 Macro Definition Documentation	900
13.158.1.1 __FFLASFFPACK_fflas_sparse_ELL_spm্ম_INL	900
13.159 ell_spmmv.inl File Reference	900
13.159.1 Macro Definition Documentation	901
13.159.1.1 __FFLASFFPACK_fflas_sparse_ELL_spmmv_INL	901
13.160 ell_utils.inl File Reference	901
13.160.1 Macro Definition Documentation	901
13.160.1.1 __FFLASFFPACK_fflas_sparse_ELL_utils_INL	901
13.161 ell_simd.h File Reference	901
13.162 ell_simd_pspmv.inl File Reference	902
13.162.1 Macro Definition Documentation	902
13.162.1.1 __FFLASFFPACK_fflas_sparse_ELL_simd_pspmv_INL	902
13.163 ell_simd_spmmv.inl File Reference	902
13.163.1 Macro Definition Documentation	903
13.163.1.1 __FFLASFFPACK_fflas_sparse_ELL_simd_spmmv_INL	903
13.164 ell_simd_utils.inl File Reference	903
13.164.1 Macro Definition Documentation	904
13.164.1.1 __FFLASFFPACK_fflas_sparse_ELL_simd_utils_INL	904
13.165 hyb_zo.h File Reference	904
13.166 hyb_zo_pspmm.inl File Reference	904
13.166.1 Macro Definition Documentation	905
13.166.1.1 __FFLASFFPACK_fflas_sparse_HYB_ZO_pspmm_INL	905
13.167 hyb_zo_pspmv.inl File Reference	905

13.167.1 Macro Definition Documentation	905
13.167.1.1 __FFLASFFPACK_fflas_sparse_HYB_ZO_pspmv_INL	905
13.168 hyb_zo_spmv.inl File Reference	905
13.168.1 Macro Definition Documentation	906
13.168.1.1 __FFLASFFPACK_fflas_sparse_HYB_ZO_spmv_INL	906
13.169 hyb_zo_spmv.inl File Reference	906
13.169.1 Macro Definition Documentation	906
13.169.1.1 __FFLASFFPACK_fflas_sparse_HYB_ZO_spmv_INL	906
13.170 hyb_zo_utils.inl File Reference	906
13.170.1 Macro Definition Documentation	906
13.170.1.1 __FFLASFFPACK_fflas_sparse_HYB_ZO_utils_INL	906
13.171 read_sparse.h File Reference	907
13.171.1 Macro Definition Documentation	907
13.171.1.1 DNS_BIN_VER	907
13.171.1.2 mask_t	908
13.172 sell.h File Reference	908
13.173 sell_pspmv.inl File Reference	908
13.173.1 Macro Definition Documentation	908
13.173.1.1 __FFLASFFPACK_fflas_sparse_sell_pspmv_INL	908
13.174 sell_spmv.inl File Reference	909
13.174.1 Macro Definition Documentation	909
13.174.1.1 __FFLASFFPACK_fflas_sparse_sell_spmv_INL	909
13.175 sell_utils.inl File Reference	909
13.175.1 Macro Definition Documentation	910
13.175.1.1 __FFLASFFPACK_fflas_sparse_sell_utils_INL	910
13.176 sparse_matrix_traits.h File Reference	910
13.177 utils.h File Reference	912
13.178 fflas_transpose.h File Reference	912
13.178.1 Detailed Description	913
13.178.2 Macro Definition Documentation	913
13.178.2.1 FFLAS_TRANSPOSE_BLOCKSIZE	913
13.178.2.2 LD	913
13.178.2.3 ST	913
13.179 ffpack.dox File Reference	913
13.180 ffpack.h File Reference	913
13.180.1 Detailed Description	922
13.180.2 Macro Definition Documentation	922
13.180.2.1 __FFLASFFPACK_FTRSTR_THRESHOLD	922
13.180.2.2 __FFLASFFPACK_FTRSSYR2K_THRESHOLD	922
13.181 ffpack.inl File Reference	922
13.181.1 Macro Definition Documentation	923
13.181.1.1 __FFLASFFPACK_ffpack_INL	923

13.182 fpack_bruhatgen.inl File Reference	923
13.182.1 Macro Definition Documentation	925
13.182.1.1 __FFLASFFPACK_ffpack_bruhatgen_inl	925
13.183 fpack_charpoly.inl File Reference	925
13.183.1 Macro Definition Documentation	925
13.183.1.1 __FFLASFFPACK_charpoly_INL	925
13.184 fpack_charpoly_danilevski.inl File Reference	925
13.184.1 Macro Definition Documentation	926
13.184.1.1 __FFLASFFPACK_ffpack_charpoly_danilveski_INL	926
13.185 fpack_charpoly_kgfast.inl File Reference	926
13.185.1 Macro Definition Documentation	926
13.185.1.1 __FFLASFFPACK_ffpack_charpoly_kgfast_INL	926
13.186 fpack_charpoly_kgfastgeneralized.inl File Reference	926
13.186.1 Macro Definition Documentation	927
13.186.1.1 __FFLASFFPACK_ffpack_charpoly_kgfastgeneralized_INL	927
13.187 fpack_charpoly_kglu.inl File Reference	927
13.187.1 Macro Definition Documentation	927
13.187.1.1 __FFLASFFPACK_ffpack_charpoly_kglu_INL	927
13.188 fpack_charpoly_mp.inl File Reference	927
13.188.1 Macro Definition Documentation	928
13.188.1.1 __FFPACK_charpoly_mp_INL	928
13.189 fpack_det_mp.inl File Reference	928
13.189.1 Macro Definition Documentation	928
13.189.1.1 __FFPACK_det_mp_INL	928
13.190 fpack_echelonforms.inl File Reference	929
13.190.1 Macro Definition Documentation	930
13.190.1.1 __FFLASFFPACK_ffpack_echelon_forms_INL	930
13.190.1.2 __FFLASFFPACK_GAUSSJORDAN_BASECASE	930
13.191 fpack_fgesv.inl File Reference	930
13.191.1 Macro Definition Documentation	930
13.191.1.1 __FFLASFFPACK_ffpack_fgesv_INL	930
13.192 fpack_fgetrs.inl File Reference	930
13.192.1 Macro Definition Documentation	931
13.192.1.1 __FFLASFFPACK_ffpack_fgetrs_INL	931
13.193 fpack_frobenius.inl File Reference	931
13.194 fpack_fsytrf.inl File Reference	932
13.194.1 Macro Definition Documentation	933
13.194.1.1 __FFLASFFPACK_ffpack_fsytrf_INL	933
13.195 fpack_ftrssyr2k.inl File Reference	933
13.195.1 Macro Definition Documentation	933
13.195.1.1 __FFLASFFPACK_ffpack_ftrssyr2k_INL	933
13.196 fpack_ftrstr.inl File Reference	933

13.196.1 Macro Definition Documentation	934
13.196.1.1 __FFLASFFPACK_ffpack_ftrstr_INL	934
13.197 fpack_ftrtr.inl File Reference	934
13.197.1 Macro Definition Documentation	934
13.197.1.1 ENABLE_ALL_CHECKINGS	934
13.197.1.2 __FFLASFFPACK_ffpack_ftrtr_INL	934
13.198 fpack_invert.inl File Reference	934
13.198.1 Macro Definition Documentation	935
13.198.1.1 __FFLASFFPACK_ffpack_invert_INL	935
13.199 fpack_krylovelim.inl File Reference	935
13.199.1 Macro Definition Documentation	935
13.199.1.1 __FFLASFFPACK_ffpack_krylovelim_INL	935
13.200 fpack_ludivine.inl File Reference	935
13.200.1 Macro Definition Documentation	936
13.200.1.1 __FFLASFFPACK_ffpack_ludivine_INL	936
13.201 fpack_ludivine_mp.inl File Reference	936
13.201.1 Macro Definition Documentation	936
13.201.1.1 __FFPACK_ludivine_mp_INL	936
13.202 fpack_minpoly.inl File Reference	937
13.202.1 Macro Definition Documentation	937
13.202.1.1 __FFLASFFPACK_ffpack_minpoly_INL	937
13.203 fpack_permutation.inl File Reference	937
13.203.1 Macro Definition Documentation	939
13.203.1.1 __FFLASFFPACK_ffpack_permutation_INL	939
13.203.1.2 FFLASFFPACK_PERM_BKSIZE	939
13.204 fpack_pluq.inl File Reference	940
13.204.1 Macro Definition Documentation	940
13.204.1.1 __FFLASFFPACK_ffpack_pluq_INL	940
13.204.1.2 CROUT	940
13.205 fpack_pluq_mp.inl File Reference	940
13.205.1 Macro Definition Documentation	941
13.205.1.1 __FFPACK_pluq_mp_INL	941
13.206 fpack_ppluq.inl File Reference	941
13.206.1 Macro Definition Documentation	941
13.206.1.1 __FFLASFFPACK_ffpack_ppluq_INL	941
13.206.1.2 __FFLAS__TRSM_READONLY	941
13.206.1.3 PBASECASE_K	941
13.207 fpack_rankprofiles.inl File Reference	942
13.207.1 Macro Definition Documentation	943
13.207.1.1 __FFLASFFPACK_ffpack_rank_profiles_INL	943
13.208 field-traits.h File Reference	943
13.208.1 Detailed Description	945

13.209 field.doxy File Reference	945
13.210 rns-double-elt.h File Reference	945
13.210.1 Detailed Description	945
13.211 rns-double-recint.inl File Reference	946
13.211.1 Macro Definition Documentation	946
13.211.1.1 __FFLASFFPACK_field_rns_double_recint_INL	946
13.212 rns-double.h File Reference	946
13.212.1 Detailed Description	947
13.212.2 Macro Definition Documentation	947
13.212.2.1 ROUND_DOWN	947
13.213 rns-double.inl File Reference	947
13.213.1 Macro Definition Documentation	947
13.213.1.1 __FFLASFFPACK_field_rns_double_INL	947
13.214 rns-integer-mod.h File Reference	947
13.214.1 Detailed Description	948
13.215 rns-integer.h File Reference	948
13.215.1 Detailed Description	949
13.216 rns.h File Reference	949
13.217 rns.inl File Reference	949
13.217.1 Macro Definition Documentation	949
13.217.1.1 __FFLASFFPACK_field_rns_INL	949
13.218 interfaces.doxy File Reference	949
13.219 fflas_c.h File Reference	949
13.219.1 Macro Definition Documentation	951
13.219.1.1 FFLAS_COMPILED	951
13.219.2 Enumeration Type Documentation	951
13.219.2.1 FFLAS_C_ORDER	951
13.219.2.2 FFLAS_C_TRANSPOSE	951
13.219.2.3 FFLAS_C_UPLO	952
13.219.2.4 FFLAS_C_DIAG	952
13.219.2.5 FFLAS_C_SIDE	952
13.219.2.6 FFLAS_C_BASE	952
13.219.3 Function Documentation	953
13.219.3.1 freducein_1_modular_double()	953
13.219.3.2 freduce_1_modular_double()	953
13.219.3.3 fnegin_1_modular_double()	953
13.219.3.4 fneg_1_modular_double()	953
13.219.3.5 fzero_1_modular_double()	953
13.219.3.6 fiszero_1_modular_double()	953
13.219.3.7 fequal_1_modular_double()	954
13.219.3.8 fassign_1_modular_double()	954
13.219.3.9 fscaln_1_modular_double()	954

13.219.3.10 fscal_1_modular_double()	954
13.219.3.11 faxpy_1_modular_double()	954
13.219.3.12 fdot_1_modular_double()	954
13.219.3.13 fswap_1_modular_double()	955
13.219.3.14 fadd_1_modular_double()	955
13.219.3.15 fsub_1_modular_double()	955
13.219.3.16 faddin_1_modular_double()	955
13.219.3.17 fsubin_1_modular_double()	955
13.219.3.18 fassign_2_modular_double()	956
13.219.3.19 fzero_2_modular_double()	956
13.219.3.20 fequal_2_modular_double()	956
13.219.3.21 fiszero_2_modular_double()	956
13.219.3.22 fidentity_2_modular_double()	956
13.219.3.23 freducein_2_modular_double()	957
13.219.3.24 freduce_2_modular_double()	957
13.219.3.25 fnegin_2_modular_double()	957
13.219.3.26 fneg_2_modular_double()	957
13.219.3.27 fscaln_2_modular_double()	957
13.219.3.28 fscal_2_modular_double()	958
13.219.3.29 faxpy_2_modular_double()	958
13.219.3.30 fmove_2_modular_double()	958
13.219.3.31 fadd_2_modular_double()	958
13.219.3.32 fsub_2_modular_double()	958
13.219.3.33 fsubin_2_modular_double()	959
13.219.3.34 faddin_2_modular_double()	959
13.219.3.35 fgemv_2_modular_double()	959
13.219.3.36 fger_2_modular_double()	959
13.219.3.37 ftrsv_2_modular_double()	960
13.219.3.38 ftrsm_3_modular_double()	960
13.219.3.39 ftrmm_3_modular_double()	960
13.219.3.40 fgemm_3_modular_double()	960
13.219.3.41 fsquare_3_modular_double()	961
13.220 fflas_L1_inst.C File Reference	961
13.220.1 Macro Definition Documentation	961
13.220.1.1 __FFLAS_L1_INST_C	961
13.220.1.2 INST_OR_DECL	961
13.220.1.3 FFLAS_FIELD [1/2]	962
13.220.1.4 FFLAS_ELT [1/6]	962
13.220.1.5 FFLAS_ELT [2/6]	962
13.220.1.6 FFLAS_ELT [3/6]	962
13.220.1.7 FFLAS_FIELD [2/2]	962
13.220.1.8 FFLAS_ELT [4/6]	962

13.220.1.9 FFLAS_ELT [5/6]	962
13.220.1.10 FFLAS_ELT [6/6]	962
13.221 fflas_L1_inst.h File Reference	962
13.221.1 Macro Definition Documentation	962
13.221.1.1 INST_OR_DECL	962
13.221.1.2 FFLAS_FIELD [1/2]	963
13.221.1.3 FFLAS_ELT [1/6]	963
13.221.1.4 FFLAS_ELT [2/6]	963
13.221.1.5 FFLAS_ELT [3/6]	963
13.221.1.6 FFLAS_FIELD [2/2]	963
13.221.1.7 FFLAS_ELT [4/6]	963
13.221.1.8 FFLAS_ELT [5/6]	963
13.221.1.9 FFLAS_ELT [6/6]	963
13.222 fflas_L1_inst_implem.inl File Reference	963
13.223 fflas_L2_inst.C File Reference	964
13.223.1 Macro Definition Documentation	965
13.223.1.1 __FFLAS_L2_INST_C	965
13.223.1.2 INST_OR_DECL	965
13.223.1.3 FFLAS_FIELD [1/2]	965
13.223.1.4 FFLAS_ELT [1/6]	965
13.223.1.5 FFLAS_ELT [2/6]	965
13.223.1.6 FFLAS_ELT [3/6]	965
13.223.1.7 FFLAS_FIELD [2/2]	965
13.223.1.8 FFLAS_ELT [4/6]	965
13.223.1.9 FFLAS_ELT [5/6]	965
13.223.1.10 FFLAS_ELT [6/6]	965
13.224 fflas_L2_inst.h File Reference	965
13.224.1 Macro Definition Documentation	966
13.224.1.1 INST_OR_DECL	966
13.224.1.2 FFLAS_FIELD [1/2]	966
13.224.1.3 FFLAS_ELT [1/6]	966
13.224.1.4 FFLAS_ELT [2/6]	966
13.224.1.5 FFLAS_ELT [3/6]	966
13.224.1.6 FFLAS_FIELD [2/2]	966
13.224.1.7 FFLAS_ELT [4/6]	966
13.224.1.8 FFLAS_ELT [5/6]	966
13.224.1.9 FFLAS_ELT [6/6]	966
13.225 fflas_L2_inst_implem.inl File Reference	966
13.226 fflas_L3_inst.C File Reference	968
13.226.1 Macro Definition Documentation	968
13.226.1.1 __FFLAS_L3_INST_C	968
13.226.1.2 INST_OR_DECL	969

13.226.1.3 FFLAS_FIELD [1/2]	969
13.226.1.4 FFLAS_ELT [1/6]	969
13.226.1.5 FFLAS_ELT [2/6]	969
13.226.1.6 FFLAS_ELT [3/6]	969
13.226.1.7 FFLAS_FIELD [2/2]	969
13.226.1.8 FFLAS_ELT [4/6]	969
13.226.1.9 FFLAS_ELT [5/6]	969
13.226.1.10 FFLAS_ELT [6/6]	969
13.227 fflas_L3_inst.h File Reference	969
13.227.1 Macro Definition Documentation	970
13.227.1.1 INST_OR_DECL	970
13.227.1.2 FFLAS_FIELD [1/2]	970
13.227.1.3 FFLAS_ELT [1/6]	970
13.227.1.4 FFLAS_ELT [2/6]	970
13.227.1.5 FFLAS_ELT [3/6]	970
13.227.1.6 FFLAS_FIELD [2/2]	970
13.227.1.7 FFLAS_ELT [4/6]	970
13.227.1.8 FFLAS_ELT [5/6]	970
13.227.1.9 FFLAS_ELT [6/6]	970
13.228 fflas_L3_inst_implem.inl File Reference	970
13.228.1 Macro Definition Documentation	971
13.228.1.1 __FFLAS__TRSM_READONLY	971
13.229 fflas_lvl1.C File Reference	971
13.229.1 Detailed Description	972
13.229.2 Function Documentation	972
13.229.2.1 freducein_1_modular_double()	972
13.229.2.2 freduce_1_modular_double()	972
13.229.2.3 fnegin_1_modular_double()	972
13.229.2.4 fneg_1_modular_double()	973
13.229.2.5 fzero_1_modular_double()	973
13.229.2.6 fiszero_1_modular_double()	973
13.229.2.7 fequal_1_modular_double()	973
13.229.2.8 fassign_1_modular_double()	973
13.229.2.9 fscaln_1_modular_double()	973
13.229.2.10 fscal_1_modular_double()	974
13.229.2.11 faxpy_1_modular_double()	974
13.229.2.12 fdot_1_modular_double()	974
13.229.2.13 fswap_1_modular_double()	974
13.229.2.14 fadd_1_modular_double()	974
13.229.2.15 fsub_1_modular_double()	975
13.229.2.16 faddn_1_modular_double()	975
13.229.2.17 fsubn_1_modular_double()	975

13.230 fflas_lvl2.C File Reference	975
13.230.1 Detailed Description	976
13.230.2 Function Documentation	976
13.230.2.1 fassign_2_modular_double()	976
13.230.2.2 fzero_2_modular_double()	977
13.230.2.3 fequal_2_modular_double()	977
13.230.2.4 fiszero_2_modular_double()	977
13.230.2.5 fidentity_2_modular_double()	977
13.230.2.6 freducein_2_modular_double()	977
13.230.2.7 freduce_2_modular_double()	977
13.230.2.8 fnegin_2_modular_double()	978
13.230.2.9 fneg_2_modular_double()	978
13.230.2.10 fscaln_2_modular_double()	978
13.230.2.11 fscal_2_modular_double()	978
13.230.2.12 faxpy_2_modular_double()	978
13.230.2.13 fmove_2_modular_double()	979
13.230.2.14 fadd_2_modular_double()	979
13.230.2.15 fsub_2_modular_double()	979
13.230.2.16 fsubin_2_modular_double()	979
13.230.2.17 faddin_2_modular_double()	980
13.230.2.18 fgemv_2_modular_double()	980
13.230.2.19 fger_2_modular_double()	980
13.230.2.20 ftrsv_2_modular_double()	980
13.231 fflas_lvl3.C File Reference	981
13.231.1 Detailed Description	981
13.231.2 Function Documentation	981
13.231.2.1 ftrsm_3_modular_double()	981
13.231.2.2 ftrmm_3_modular_double()	982
13.231.2.3 fgemm_3_modular_double()	982
13.231.2.4 fsquare_3_modular_double()	982
13.232 fflas_sparse.C File Reference	982
13.232.1 Detailed Description	982
13.233 ffpack.C File Reference	983
13.233.1 Detailed Description	986
13.233.2 Function Documentation	986
13.233.2.1 LAPACKPerm2MathPerm()	986
13.233.2.2 MathPerm2LAPACKPerm()	986
13.233.2.3 MatrixApplyS_modular_double()	986
13.233.2.4 PermApplyS_double()	987
13.233.2.5 MatrixApplyT_modular_double()	987
13.233.2.6 PermApplyT_double()	987
13.233.2.7 composePermutationsLLM()	987

13.233.2.8 composePermutationsLLL()	987
13.233.2.9 composePermutationsMLM()	988
13.233.2.10 cyclic_shift_mathPerm()	988
13.233.2.11 cyclic_shift_row_modular_double()	988
13.233.2.12 cyclic_shift_col_modular_double()	988
13.233.2.13 applyP_modular_double()	988
13.233.2.14 fgetrsin_modular_double()	988
13.233.2.15 fgetrsv_modular_double()	989
13.233.2.16 fgesvin_modular_double()	989
13.233.2.17 fgesv_modular_double()	989
13.233.2.18 ftrtri_modular_double()	990
13.233.2.19 trinv_left_modular_double()	990
13.233.2.20 ftrtrm_modular_double()	990
13.233.2.21 PLUQ_modular_double()	990
13.233.2.22 LUdivine_modular_double()	990
13.233.2.23 ColumnEchelonForm_modular_double()	991
13.233.2.24 RowEchelonForm_modular_double()	991
13.233.2.25 ReducedColumnEchelonForm_modular_double()	991
13.233.2.26 ReducedRowEchelonForm_modular_double()	991
13.233.2.27 ColumnEchelonForm_modular_float()	992
13.233.2.28 RowEchelonForm_modular_float()	992
13.233.2.29 ReducedColumnEchelonForm_modular_float()	992
13.233.2.30 ReducedRowEchelonForm_modular_float()	992
13.233.2.31 ColumnEchelonForm_modular_int32_t()	992
13.233.2.32 RowEchelonForm_modular_int32_t()	993
13.233.2.33 ReducedColumnEchelonForm_modular_int32_t()	993
13.233.2.34 ReducedRowEchelonForm_modular_int32_t()	993
13.233.2.35 pColumnEchelonForm_modular_double()	993
13.233.2.36 pRowEchelonForm_modular_double()	994
13.233.2.37 pReducedColumnEchelonForm_modular_double()	994
13.233.2.38 pReducedRowEchelonForm_modular_double()	994
13.233.2.39 pColumnEchelonForm_modular_float()	994
13.233.2.40 pRowEchelonForm_modular_float()	995
13.233.2.41 pReducedColumnEchelonForm_modular_float()	995
13.233.2.42 pReducedRowEchelonForm_modular_float()	995
13.233.2.43 pColumnEchelonForm_modular_int32_t()	995
13.233.2.44 pRowEchelonForm_modular_int32_t()	996
13.233.2.45 pReducedColumnEchelonForm_modular_int32_t()	996
13.233.2.46 pReducedRowEchelonForm_modular_int32_t()	996
13.233.2.47 Invertin_modular_double()	996
13.233.2.48 Invert_modular_double()	996
13.233.2.49 Invert2_modular_double()	997

13.233.2.50 KrylovElim_modular_double()	997
13.233.2.51 SpecRankProfile_modular_double()	997
13.233.2.52 Rank_modular_double()	997
13.233.2.53 IsSingular_modular_double()	997
13.233.2.54 Det_modular_double()	998
13.233.2.55 Solve_modular_double()	998
13.233.2.56 solveLB_modular_double()	998
13.233.2.57 solveLB2_modular_double()	998
13.233.2.58 RandomNullSpaceVector_modular_double()	999
13.233.2.59 NullSpaceBasis_modular_double()	999
13.233.2.60 RowRankProfile_modular_double()	999
13.233.2.61 ColumnRankProfile_modular_double()	999
13.233.2.62 RankProfileFromLU()	999
13.233.2.63 LeadingSubmatrixRankProfiles()	1000
13.233.2.64 RowRankProfileSubmatrixIndices_modular_double()	1000
13.233.2.65 ColRankProfileSubmatrixIndices_modular_double()	1000
13.233.2.66 RowRankProfileSubmatrix_modular_double()	1000
13.233.2.67 ColRankProfileSubmatrix_modular_double()	1000
13.233.2.68 getTriangular_modular_double()	1001
13.233.2.69 getTriangularin_modular_double()	1001
13.233.2.70 getEchelonForm_modular_double()	1001
13.233.2.71 getEchelonFormin_modular_double()	1001
13.233.2.72 getEchelonTransform_modular_double()	1002
13.233.2.73 getReducedEchelonForm_modular_double()	1002
13.233.2.74 getReducedEchelonFormin_modular_double()	1002
13.233.2.75 getReducedEchelonTransform_modular_double()	1002
13.233.2.76 PLUQtoEchelonPermutation()	1003
13.234 fpack_c.h File Reference	1003
13.234.1 Macro Definition Documentation	1006
13.234.1.1 FFPACK_COMPILED	1006
13.234.2 Enumeration Type Documentation	1006
13.234.2.1 FFLAS_C_ORDER	1006
13.234.2.2 FFLAS_C_TRANSPOSE	1006
13.234.2.3 FFLAS_C_UPLO	1006
13.234.2.4 FFLAS_C_DIAG	1007
13.234.2.5 FFLAS_C_SIDE	1007
13.234.2.6 FFPACK_C_LU_TAG	1007
13.234.2.7 FFPACK_C_CHARPOLY_TAG	1007
13.234.2.8 FFPACK_C_MINPOLY_TAG	1007
13.234.3 Function Documentation	1008
13.234.3.1 LAPACKPerm2MathPerm()	1008
13.234.3.2 MathPerm2LAPACKPerm()	1008

13.234.3.3 MatrixApplyS_modular_double()	1008
13.234.3.4 PermApplyS_double()	1008
13.234.3.5 MatrixApplyT_modular_double()	1008
13.234.3.6 PermApplyT_double()	1009
13.234.3.7 composePermutationsLLM()	1009
13.234.3.8 composePermutationsLLL()	1009
13.234.3.9 composePermutationsMLM()	1009
13.234.3.10 cyclic_shift_mathPerm()	1009
13.234.3.11 cyclic_shift_row_modular_double()	1009
13.234.3.12 cyclic_shift_col_modular_double()	1009
13.234.3.13 applyP_modular_double()	1010
13.234.3.14 fgetrsin_modular_double()	1010
13.234.3.15 fgetrs_modular_double()	1010
13.234.3.16 fgesvin_modular_double()	1011
13.234.3.17 fgesv_modular_double()	1011
13.234.3.18 ftrtri_modular_double()	1011
13.234.3.19 trinv_left_modular_double()	1011
13.234.3.20 ftrtrm_modular_double()	1011
13.234.3.21 PLUQ_modular_double()	1012
13.234.3.22 LUdivine_modular_double()	1012
13.234.3.23 LUdivine_small_modular_double()	1012
13.234.3.24 LUdivine_gauss_modular_double()	1012
13.234.3.25 ColumnEchelonForm_modular_double()	1013
13.234.3.26 RowEchelonForm_modular_double()	1013
13.234.3.27 ColumnEchelonForm_modular_float()	1013
13.234.3.28 RowEchelonForm_modular_float()	1013
13.234.3.29 ColumnEchelonForm_modular_int32_t()	1014
13.234.3.30 RowEchelonForm_modular_int32_t()	1014
13.234.3.31 ReducedColumnEchelonForm_modular_double()	1014
13.234.3.32 ReducedRowEchelonForm_modular_double()	1014
13.234.3.33 ReducedColumnEchelonForm_modular_float()	1014
13.234.3.34 ReducedRowEchelonForm_modular_float()	1015
13.234.3.35 ReducedColumnEchelonForm_modular_int32_t()	1015
13.234.3.36 ReducedRowEchelonForm_modular_int32_t()	1015
13.234.3.37 ReducedRowEchelonForm2_modular_double()	1015
13.234.3.38 REF_modular_double()	1016
13.234.3.39 Invertin_modular_double()	1016
13.234.3.40 Invert_modular_double()	1016
13.234.3.41 Invert2_modular_double()	1016
13.234.3.42 KrylovElim_modular_double()	1016
13.234.3.43 SpecRankProfile_modular_double()	1017
13.234.3.44 Rank_modular_double()	1017

13.234.3.45	IsSingular_modular_double()	1017
13.234.3.46	Det_modular_double()	1017
13.234.3.47	Solve_modular_double()	1017
13.234.3.48	solveLB_modular_double()	1018
13.234.3.49	solveLB2_modular_double()	1018
13.234.3.50	RandomNullSpaceVector_modular_double()	1018
13.234.3.51	NullSpaceBasis_modular_double()	1018
13.234.3.52	RowRankProfile_modular_double()	1019
13.234.3.53	ColumnRankProfile_modular_double()	1019
13.234.3.54	RankProfileFromLU()	1019
13.234.3.55	LeadingSubmatrixRankProfiles()	1019
13.234.3.56	RowRankProfileSubmatrixIndices_modular_double()	1019
13.234.3.57	ColRankProfileSubmatrixIndices_modular_double()	1020
13.234.3.58	RowRankProfileSubmatrix_modular_double()	1020
13.234.3.59	ColRankProfileSubmatrix_modular_double()	1020
13.234.3.60	getTriangular_modular_double()	1020
13.234.3.61	getTriangularin_modular_double()	1021
13.234.3.62	getEchelonForm_modular_double()	1021
13.234.3.63	getEchelonFormin_modular_double()	1021
13.234.3.64	getEchelonTransform_modular_double()	1021
13.234.3.65	getReducedEchelonForm_modular_double()	1022
13.234.3.66	getReducedEchelonFormin_modular_double()	1022
13.234.3.67	getReducedEchelonTransform_modular_double()	1022
13.234.3.68	PLUQtoEchelonPermutation()	1022
13.235	ffpack_inst.C File Reference	1023
13.235.1	Macro Definition Documentation	1023
13.235.1.1	__FFPACK_INST_C	1023
13.235.1.2	FFLAS_COMPILED	1023
13.235.1.3	INST_OR_DECL	1023
13.235.1.4	FFLAS_FIELD [1/2]	1023
13.235.1.5	FFLAS_ELT [1/6]	1023
13.235.1.6	FFLAS_ELT [2/6]	1023
13.235.1.7	FFLAS_ELT [3/6]	1023
13.235.1.8	FFLAS_FIELD [2/2]	1023
13.235.1.9	FFLAS_ELT [4/6]	1024
13.235.1.10	FFLAS_ELT [5/6]	1024
13.235.1.11	FFLAS_ELT [6/6]	1024
13.236	ffpack_inst.h File Reference	1024
13.236.1	Macro Definition Documentation	1024
13.236.1.1	FFLAS_COMPILED	1024
13.236.1.2	INST_OR_DECL	1024
13.236.1.3	FFLAS_FIELD [1/2]	1024

13.236.1.4 FFLAS_ELT [1/6]	1024
13.236.1.5 FFLAS_ELT [2/6]	1024
13.236.1.6 FFLAS_ELT [3/6]	1024
13.236.1.7 FFLAS_FIELD [2/2]	1025
13.236.1.8 FFLAS_ELT [4/6]	1025
13.236.1.9 FFLAS_ELT [5/6]	1025
13.236.1.10 FFLAS_ELT [6/6]	1025
13.237 ffpack_inst_implem.inl File Reference	1025
13.238 blockcuts.inl File Reference	1028
13.238.1 Macro Definition Documentation	1029
13.238.1.1 __FFLASFFPACK_fflas_blockcuts_INL	1029
13.238.1.2 __FFLASFFPACK_MINBLOCKCUTS	1029
13.239 fflas_plevel1.h File Reference	1030
13.240 kaapi_routines.inl File Reference	1030
13.240.1 Macro Definition Documentation	1030
13.240.1.1 __FFLASFFPACK_KAAPI_ROUTINES_INL	1030
13.241 parallel.h File Reference	1030
13.241.1 Macro Definition Documentation	1031
13.241.1.1 __FFLASFFPACK_SEQUENTIAL	1031
13.241.1.2 index_t	1031
13.241.1.3 TASK	1031
13.241.1.4 WAIT	1031
13.241.1.5 CHECK_DEPENDENCIES	1031
13.241.1.6 BARRIER	1031
13.241.1.7 PAR_BLOCK	1032
13.241.1.8 SYNCH_GROUP	1032
13.241.1.9 THREAD_INDEX	1032
13.241.1.10 NUM_THREADS	1032
13.241.1.11 SET_THREADS	1032
13.241.1.12 MAX_THREADS	1032
13.241.1.13 READ	1032
13.241.1.14 WRITE	1032
13.241.1.15 READWRITE	1032
13.241.1.16 CONSTREFERENCE	1032
13.241.1.17 VALUE	1032
13.241.1.18 BEGIN_PARALLEL_MAIN	1032
13.241.1.19 END_PARALLEL_MAIN	1032
13.241.1.20 FORBLOCK1D	1033
13.241.1.21 FOR1D	1033
13.241.1.22 PARFORBLOCK1D	1033
13.241.1.23 PARFOR1D	1033
13.241.1.24 FORBLOCK2D	1033

13.241.1.25 FOR2D	1033
13.241.1.26 PARFORBLOCK2D	1034
13.241.1.27 PARFOR2D	1034
13.241.1.28 COMMA	1034
13.241.1.29 MODE	1034
13.241.1.30 RETURNPARAM	1034
13.241.1.31 NUMARGS	1034
13.241.1.32 PP_NARG_	1034
13.241.1.33 PP_ARG_N	1034
13.241.1.34 PP_RSEQ_N	1036
13.241.1.35 NOSPLIT	1036
13.241.1.36 splitting_0	1036
13.241.1.37 splitting_1	1036
13.241.1.38 splitting_2	1036
13.241.1.39 splitting_3	1036
13.241.1.40 splitt	1036
13.241.1.41 SPLITTER	1036
13.242 pfgemm_variants.inl File Reference	1037
13.243 pfgemv.inl File Reference	1037
13.244 align-allocator.h File Reference	1038
13.245 args-parser.h File Reference	1038
13.245.1 Macro Definition Documentation	1039
13.245.1.1 TYPE_BOOL	1039
13.245.1.2 END_OF_ARGUMENTS	1039
13.245.1.3 type_integer	1039
13.245.2 Enumeration Type Documentation	1039
13.245.2.1 ArgumentType	1039
13.245.3 Function Documentation	1039
13.245.3.1 printHelpMessage()	1039
13.245.3.2 findArgument()	1039
13.245.3.3 getListArgs()	1039
13.246 bit_manipulation.h File Reference	1040
13.246.1 Macro Definition Documentation	1040
13.246.1.1 __has_builtin	1040
13.246.2 Function Documentation	1040
13.246.2.1 clz() [1/2]	1040
13.246.2.2 clz() [2/2]	1040
13.246.2.3 ctz() [1/2]	1040
13.246.2.4 ctz() [2/2]	1040
13.247 cast.h File Reference	1041
13.248 debug.h File Reference	1041
13.248.1 Detailed Description	1041

13.248.2 Macro Definition Documentation	1041
13.248.2.1 FFLASFFPACK_check	1041
13.248.2.2 FFLASFFPACK_abort	1042
13.249 fflas_intrinsic.h File Reference	1042
13.250 fflas_io.h File Reference	1042
13.251 fflas_memory.h File Reference	1043
13.252 fflas_randommatrix.h File Reference	1043
13.253 flimits.h File Reference	1045
13.253.1 Function Documentation	1046
13.253.1.1 in_range() [1/3]	1046
13.253.1.2 in_range() [2/3]	1046
13.253.1.3 in_range() [3/3]	1046
13.254 Matio.h File Reference	1046
13.254.1 Function Documentation	1047
13.254.1.1 read_field()	1047
13.254.1.2 write_field()	1047
13.255 test-utils.h File Reference	1047
13.256 timer.h File Reference	1048
13.257 cblas.C File Reference	1048
13.257.1 Macro Definition Documentation	1048
13.257.1.1 __FFLASFFPACK_CONFIGURATION	1048
13.257.1.2 __FFLASFFPACK_HAVE_CBLAS	1048
13.257.2 Function Documentation	1048
13.257.2.1 main()	1048
13.258 clapack.C File Reference	1048
13.258.1 Macro Definition Documentation	1049
13.258.1.1 __FFLASFFPACK_CONFIGURATION	1049
13.258.1.2 __FFLASFFPACK_HAVE_LAPACK	1049
13.258.1.3 __FFLASFFPACK_HAVE_CLAPACK	1049
13.258.2 Function Documentation	1049
13.258.2.1 main()	1049
13.259 cuda.C File Reference	1049
13.259.1 Function Documentation	1049
13.259.1.1 main()	1049
13.260 fblas.C File Reference	1049
13.260.1 Macro Definition Documentation	1050
13.260.1.1 __FFLASFFPACK_CONFIGURATION	1050
13.260.2 Function Documentation	1050
13.260.2.1 dgemm_()	1050
13.260.2.2 main()	1050
13.261 lapack.C File Reference	1050
13.261.1 Macro Definition Documentation	1050

13.261.1.1 __FFLASFFPACK_CONFIGURATION	1050
13.261.1.2 __FFLASFFPACK_HAVE_LAPACK	1050
13.261.2 Function Documentation	1050
13.261.2.1 main()	1050
13.262 regression-check.C File Reference	1051
13.262.1 Function Documentation	1051
13.262.1.1 check1()	1051
13.262.1.2 check2()	1051
13.262.1.3 check3()	1051
13.262.1.4 check4()	1051
13.262.1.5 checkZeroDimCharpoly()	1051
13.262.1.6 checkZeroDimMinPoly()	1051
13.262.1.7 gf2ModularBalanced()	1051
13.262.1.8 main()	1051
13.263 test-charpoly-check.C File Reference	1052
13.263.1 Macro Definition Documentation	1052
13.263.1.1 ENABLE_CHECKER_charpoly	1052
13.263.1.2 TIME_CHECKER_CHARPOLY	1052
13.263.2 Function Documentation	1052
13.263.2.1 printPolynomial()	1052
13.263.2.2 main()	1052
13.264 test-charpoly.C File Reference	1052
13.264.1 Function Documentation	1053
13.264.1.1 launch_test()	1053
13.264.1.2 run_with_field()	1053
13.264.1.3 main()	1053
13.265 test-compressQ.C File Reference	1053
13.265.1 Typedef Documentation	1054
13.265.1.1 Field	1054
13.265.2 Function Documentation	1054
13.265.2.1 printvect()	1054
13.265.2.2 main()	1054
13.266 test-det-check.C File Reference	1054
13.266.1 Macro Definition Documentation	1054
13.266.1.1 ENABLE_CHECKER_Det	1054
13.266.1.2 TIME_CHECKER_Det	1055
13.266.2 Function Documentation	1055
13.266.2.1 main()	1055
13.267 test-det.C File Reference	1055
13.267.1 Function Documentation	1055
13.267.1.1 test_det()	1055
13.267.1.2 main()	1055

13.268 test-echelon.C File Reference	1055
13.268.1 Macro Definition Documentation	1056
13.268.1.1 __FFLASFFPACK_SEQUENTIAL	1056
13.268.1.2 __FFLASFFPACK_GAUSSJORDAN_BASECASE	1056
13.268.1.3 __FFLASFFPACK_PLUQ_THRESHOLD	1056
13.268.2 Function Documentation	1056
13.268.2.1 test_colechelon()	1056
13.268.2.2 test_rowechelon()	1057
13.268.2.3 test_redcoechelon()	1057
13.268.2.4 test_redrowechelon()	1057
13.268.2.5 run_with_field()	1057
13.268.2.6 main()	1058
13.269 test-fadd.C File Reference	1058
13.269.1 Function Documentation	1058
13.269.1.1 test_fadd()	1058
13.269.1.2 test_faddin()	1058
13.269.1.3 test_fsub()	1058
13.269.1.4 test_fsubin()	1059
13.269.1.5 main()	1059
13.270 test-fdot.C File Reference	1059
13.270.1 Macro Definition Documentation	1059
13.270.1.1 ENABLE_ALL_CHECKINGS	1059
13.270.2 Function Documentation	1060
13.270.2.1 check_fdot()	1060
13.270.2.2 run_with_field()	1060
13.270.2.3 run_with_Integer()	1060
13.270.2.4 main()	1060
13.271 test-fgemm-check.C File Reference	1060
13.271.1 Macro Definition Documentation	1061
13.271.1.1 ENABLE_ALL_CHECKINGS	1061
13.271.2 Function Documentation	1061
13.271.2.1 launch_MM_dispatch()	1061
13.271.2.2 run_with_field()	1061
13.271.2.3 main()	1061
13.272 test-fgemm.C File Reference	1061
13.272.1 Macro Definition Documentation	1062
13.272.1.1 ENABLE_CHECKER_fgemm	1062
13.272.2 Function Documentation	1062
13.272.2.1 check_MM()	1062
13.272.2.2 launch_MM()	1062
13.272.2.3 launch_MM_dispatch()	1063
13.272.2.4 run_with_field()	1063

13.272.2.5 main()	1063
13.273 test-fgemv.C File Reference	1064
13.273.1 Function Documentation	1064
13.273.1.1 check_MV()	1064
13.273.1.2 launch_MV()	1064
13.273.1.3 launch_MV_dispatch()	1065
13.273.1.4 run_with_field()	1065
13.273.1.5 main()	1065
13.274 test-fger.C File Reference	1065
13.274.1 Macro Definition Documentation	1066
13.274.1.1 TIME	1066
13.274.2 Function Documentation	1066
13.274.2.1 check_fger()	1066
13.274.2.2 launch_fger()	1066
13.274.2.3 launch_fger_dispatch()	1066
13.274.2.4 run_with_field()	1067
13.274.2.5 main()	1067
13.275 test-fgesv.C File Reference	1067
13.275.1 Function Documentation	1068
13.275.1.1 test_square_fgesv()	1068
13.275.1.2 test_rect_fgesv()	1068
13.275.1.3 run_with_field()	1068
13.275.1.4 main()	1068
13.276 test-finit.C File Reference	1068
13.276.1 Function Documentation	1069
13.276.1.1 test_freduce()	1069
13.276.1.2 run_with_field()	1069
13.276.1.3 main()	1069
13.277 test-fscal.C File Reference	1069
13.277.1 Function Documentation	1070
13.277.1.1 test_fscal() [1/2]	1070
13.277.1.2 test_fscal() [2/2]	1070
13.277.1.3 test_fscalin() [1/2]	1070
13.277.1.4 test_fscalin() [2/2]	1070
13.277.1.5 main()	1071
13.278 test-fsyr2k.C File Reference	1071
13.278.1 Macro Definition Documentation	1071
13.278.1.1 ENABLE_ALL_CHECKINGS	1071
13.278.2 Function Documentation	1071
13.278.2.1 check_fsyr2k()	1071
13.278.2.2 run_with_field()	1072
13.278.2.3 main()	1072

13.279 test-fsyrr.C File Reference	1072
13.279.1 Macro Definition Documentation	1073
13.279.1.1 ENABLE_ALL_CHECKINGS	1073
13.279.2 Function Documentation	1073
13.279.2.1 check_fsyrr()	1073
13.279.2.2 check_fsyrr_diag()	1073
13.279.2.3 check_fsyrr_bkdiag()	1073
13.279.2.4 check_computeS1S2()	1073
13.279.2.5 run_with_field()	1074
13.279.2.6 main()	1074
13.280 test-fsytrf.C File Reference	1074
13.280.1 Function Documentation	1074
13.280.1.1 operator<<()	1074
13.280.1.2 test_RPM_fsytrf()	1075
13.280.1.3 test_generic_fsytrf()	1075
13.280.1.4 run_with_field()	1075
13.280.1.5 main()	1075
13.281 test-frm.C File Reference	1075
13.281.1 Macro Definition Documentation	1076
13.281.1.1 __FFLASFFPACK_SEQUENTIAL	1076
13.281.2 Function Documentation	1076
13.281.2.1 check_frm()	1076
13.281.2.2 run_with_field()	1076
13.281.2.3 main()	1076
13.282 test-frm.C File Reference	1076
13.282.1 Macro Definition Documentation	1077
13.282.1.1 __FFLASFFPACK_SEQUENTIAL	1077
13.282.1.2 ENABLE_ALL_CHECKINGS	1077
13.282.2 Function Documentation	1077
13.282.2.1 check_frm()	1077
13.282.2.2 run_with_field()	1077
13.282.2.3 main()	1078
13.283 test-frm-check.C File Reference	1078
13.283.1 Macro Definition Documentation	1078
13.283.1.1 ENABLE_ALL_CHECKINGS	1078
13.283.2 Function Documentation	1078
13.283.2.1 main()	1078
13.284 test-frm.C File Reference	1078
13.284.1 Macro Definition Documentation	1079
13.284.1.1 __FFLASFFPACK_SEQUENTIAL	1079
13.284.1.2 ENABLE_ALL_CHECKINGS	1079
13.284.2 Function Documentation	1079

13.284.2.1 check_ftsm()	1079
13.284.2.2 run_with_field()	1079
13.284.2.3 main()	1079
13.285 test-ftssyr2k.C File Reference	1079
13.285.1 Macro Definition Documentation	1080
13.285.1.1 ENABLE_ALL_CHECKINGS	1080
13.285.2 Function Documentation	1080
13.285.2.1 check_ftssyr2k()	1080
13.285.2.2 run_with_field()	1080
13.285.2.3 main()	1080
13.286 test-ftstr.C File Reference	1081
13.286.1 Macro Definition Documentation	1081
13.286.1.1 ENABLE_ALL_CHECKINGS	1081
13.286.2 Function Documentation	1081
13.286.2.1 check_ftstr()	1081
13.286.2.2 run_with_field()	1081
13.286.2.3 main()	1082
13.287 test-ftsv.C File Reference	1082
13.287.1 Macro Definition Documentation	1082
13.287.1.1 __FFLASFFPACK_SEQUENTIAL	1082
13.287.1.2 ENABLE_ALL_CHECKINGS	1082
13.287.2 Function Documentation	1082
13.287.2.1 check_ftsv()	1082
13.287.2.2 run_with_field()	1083
13.287.2.3 main()	1083
13.288 test-fttri.C File Reference	1083
13.288.1 Macro Definition Documentation	1083
13.288.1.1 __FFLASFFPACK_SEQUENTIAL	1083
13.288.1.2 ENABLE_ALL_CHECKINGS	1083
13.288.2 Function Documentation	1083
13.288.2.1 check_fttri()	1083
13.288.2.2 run_with_field()	1084
13.288.2.3 main()	1084
13.289 test-interfaces-c.c File Reference	1084
13.289.1 Function Documentation	1084
13.289.1.1 main()	1084
13.290 test-invert-check.C File Reference	1084
13.290.1 Macro Definition Documentation	1085
13.290.1.1 ENABLE_ALL_CHECKINGS	1085
13.290.2 Function Documentation	1085
13.290.2.1 main()	1085
13.291 test-io.C File Reference	1085

13.291.1 Function Documentation	1085
13.291.1.1 run_with_field()	1085
13.291.1.2 main()	1085
13.292 test-lu.C File Reference	1086
13.292.1 Macro Definition Documentation	1086
13.292.1.1 BASECASE_K	1086
13.292.1.2 __FFLASFFPACK_SEQUENTIAL	1087
13.292.1.3 __LUDIVINE_CUTOFF	1087
13.292.2 Function Documentation	1087
13.292.2.1 test_LUdivine()	1087
13.292.2.2 verifPLUQ()	1087
13.292.2.3 test_pluq()	1088
13.292.2.4 launch_test()	1088
13.292.2.5 run_with_field()	1089
13.292.2.6 main()	1089
13.292.3 Variable Documentation	1089
13.292.3.1 tperm	1089
13.292.3.2 tgemm	1089
13.292.3.3 tBC	1089
13.292.3.4 ttrsm	1089
13.292.3.5 trest	1089
13.292.3.6 timtot	1089
13.292.3.7 mvcnt	1089
13.293 test-maxdelayeddim.C File Reference	1090
13.293.1 Macro Definition Documentation	1090
13.293.1.1 MAX_WITH_SIZE_T	1090
13.293.2 Function Documentation	1090
13.293.2.1 test()	1090
13.293.2.2 main()	1090
13.294 test-minpoly.C File Reference	1090
13.294.1 Function Documentation	1091
13.294.1.1 check_minpoly()	1091
13.294.1.2 run_with_field()	1091
13.294.1.3 main()	1091
13.295 test-multifile1.C File Reference	1091
13.296 test-multifile2.C File Reference	1091
13.296.1 Function Documentation	1091
13.296.1.1 main()	1091
13.297 test-nullspace.C File Reference	1092
13.297.1 Function Documentation	1092
13.297.1.1 checkingMessage()	1092
13.297.1.2 readOrRandomMatrixWithRankAndRandomRPM()	1092

13.297.1.3 test_nullspace()	1092
13.297.1.4 run_with_field()	1093
13.297.1.5 main()	1093
13.298 test-permutations.C File Reference	1093
13.298.1 Function Documentation	1093
13.298.1.1 checkMonotonicApplyP()	1093
13.298.1.2 main()	1093
13.298.2 Variable Documentation	1094
13.298.2.1 tperm	1094
13.298.2.2 tgemm	1094
13.298.2.3 tBC	1094
13.298.2.4 ttrsm	1094
13.298.2.5 trest	1094
13.298.2.6 timtot	1094
13.299 test-pluq-check.C File Reference	1094
13.299.1 Macro Definition Documentation	1094
13.299.1.1 ENABLE_ALL_CHECKINGS	1094
13.299.2 Function Documentation	1094
13.299.2.1 main()	1094
13.300 test-quasisep.C File Reference	1095
13.300.1 Function Documentation	1095
13.300.1.1 test_BruhatGenerator()	1095
13.300.1.2 launch_test()	1095
13.300.1.3 testLTQSRPM()	1095
13.300.1.4 run_with_field()	1096
13.300.1.5 main()	1096
13.301 test-rankprofiles.C File Reference	1096
13.301.1 Macro Definition Documentation	1096
13.301.1.1 __FFLASFFPACK_SEQUENTIAL	1096
13.301.2 Function Documentation	1096
13.301.2.1 run_with_field()	1096
13.301.2.2 main()	1097
13.302 test-rpm.C File Reference	1097
13.302.1 Function Documentation	1097
13.302.1.1 checkRPM()	1097
13.302.1.2 checkSymmetricRPM()	1097
13.302.1.3 main()	1097
13.303 test-simd.C File Reference	1097
13.303.1 Macro Definition Documentation	1099
13.303.1.1 _TEST_ONE	1099
13.303.1.2 TEST_ONE_OP	1099
13.303.1.3 TEST_ONE_OP_WZ	1099

13.303.1.4 TEST_IMPL	1099
13.303.2 Function Documentation	1099
13.303.2.1 check_eq() [1/2]	1099
13.303.2.2 check_eq() [2/2]	1100
13.303.2.3 cmp()	1100
13.303.2.4 eval_func_on_array() [1/3]	1100
13.303.2.5 eval_func_on_array() [2/3]	1100
13.303.2.6 eval_func_on_array() [3/3]	1100
13.303.2.7 operator<<()	1100
13.303.2.8 test_impl_base() [1/2]	1100
13.303.2.9 test_impl_base() [2/2]	1100
13.303.2.10 test_impl()	1100
13.303.2.11 main()	1101
13.304 test-solve.C File Reference	1101
13.304.1 Function Documentation	1101
13.304.1.1 check_solve()	1101
13.304.1.2 run_with_field()	1101
13.304.1.3 main()	1101
13.305 test-storage-transpose.C File Reference	1101
13.305.1 Function Documentation	1102
13.305.1.1 main()	1102
13.306 101-fgemv.C File Reference	1102
13.306.1 Function Documentation	1102
13.306.1.1 main()	1102
13.307 2x2-fgemv.C File Reference	1102
13.307.1 Function Documentation	1103
13.307.1.1 main()	1103
13.308 2x2-ftsv.C File Reference	1103
13.308.1 Function Documentation	1103
13.308.1.1 main()	1103
13.309 2x2-pluq.C File Reference	1103
13.309.1 Function Documentation	1103
13.309.1.1 main()	1103
13.310 fflas-101_1.C File Reference	1104
13.310.1 Function Documentation	1104
13.310.1.1 main()	1104
13.311 fflas-101_3.C File Reference	1104
13.311.1 Function Documentation	1104
13.311.1.1 main()	1104
13.312 fflas_101.C File Reference	1104
13.312.1 Function Documentation	1105
13.312.1.1 main()	1105

13.313 fflas_101_lvl1.C File Reference	1105
13.313.1 Function Documentation	1105
13.313.1.1 main()	1105
13.314 ffpack-fgesv.C File Reference	1105
13.314.1 Function Documentation	1105
13.314.1.1 main()	1105
13.315 ffpack-solve.C File Reference	1105
13.315.1 Function Documentation	1106
13.315.1.1 main()	1106
Index	1107

Chapter 1

FFLAS-FFPACK Documentation.

1.1 Introduction

FFLAS-FFPACK is a LGPL-2.1+ source code library for basic linear algebra operations over a finite field. It is inspired by BLAS interface (Basic Linear Algebra Subprograms) and the LAPACK library for numerical linear algebra, and shares part of their design. Yet it differs in many aspects due to the specifics of computing over a finite field:

- it is generic with respect to the finite field, so as to accomodate a large variety of field sizes and implementations;
- it is a pure source code library, to be included and compiled in the user's software. Its build system is only used for tests and benchmarks.

1.2 Goals

1.3 Design

1.4 Using FFLAS-FFPACK.

- [Copying and Licence](#).
- [Tutorial](#). This is a brief introduction to FFLAS-[FFPACK](#) capabilities.
- [Configuring and Installing FFLAS-FFPACK](#). Explains how to configure/install from sources or from the latest svn version.
- [Architecture of the library](#).. Describes how FFLAS-FFPACK is organized
- [Documentation for Users](#). If everything around is blue, then you are reading the lighter, user-oriented, documentation.
- [Documentation for Developers](#). If everything around is green, then you can get to everything (not necessarily yet) documented.

1.5 Contributing to fflas-ffpack, getting assistance.

Version

2.5.0

1.6 Copying and Licence

The FFLAS-FFPACK library is licensed under the terms of the GNU LGPL v2.1 or later.

See <https://www.gnu.org/licenses/lgpl-2.1.html>

1.7 Tutorial

no doc.

1.8 Configuring and Installing FFLAS-FFPACK

FFLAS-FFPACK is a header-only package.

Howver configuration process can be tweaked a lot. Configure looks for BLAS routines and [Givaro](#) library which are both mandatory dependencies. See the output of `./configure -help` for information about the LAPACK/↔ BLAS discovering strategies.

1.9 Architecture of the library.

no doc.

Chapter 2

Bug List

Global **DOUBLE_TO_FLOAT_CROSSOVER**

to be benchmarked.

Global **FFLAS::details::pack_lhs** (int64_t *XX, const int64_t *X, size_t ldx, size_t rows, size_t cols)

this is fassign

this is fassign

Global **FFLAS::details::pack_rhs** (int64_t *XX, const int64_t *X, size_t ldx, size_t rows, size_t cols)

this is fassign

this is fassign

Global **FFLAS::fconvert** (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t n, FFLAS_ELT *X, const size_t incX, const FFLAS_ELT *Y, const size_t incY)

use cblas_(d)scal when possible

Global **FFLAS::fconvert** (const Field &F, const size_t n, OtherElement_ptr X, const size_t incX, typename Field::ConstElement_ptr Y, const size_t incY)

use cblas_(d)scal when possible

Global **FFLAS::finit** (const Field &F, const size_t n, const OtherElement_ptr Y, const size_t incY, typename Field::Element_ptr X, const size_t incX)

use cblas_(d)scal when possible

Global **FFLAS::finit** (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t n, const FFLAS_ELT *Y, const size_t incY, FFLAS_ELT *X, const size_t incX)

use cblas_(d)scal when possible

Global **FFLAS::fneg** (const Field &F, const size_t n, typename Field::ConstElement_ptr Y, const size_t incY, typename Field::Element_ptr X, const size_t incX)

use cblas_(d)scal when possible

Global **FFLAS::fneg** (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t n, const FFLAS_ELT *Y, const size_t incY, FFLAS_ELT *X, const size_t incX)

use cblas_(d)scal when possible

Global **FFLAS::fnegin** (const Field &F, const size_t n, typename Field::Element_ptr X, const size_t incX)

use cblas_(d)scal when possible

Global **FFLAS::fnegin** (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t n, FFLAS_ELT *X, const size_t incX)

use cblas_(d)scal when possible

Global **FFLAS::freduce** (const Field &F, const size_t n, typename Field::ConstElement_ptr Y, const size_t incY, typename Field::Element_ptr X, const size_t incX)

use cblas_(d)scal when possible

- Global **FFLAS::reduce** (const Field &F, const size_t n, typename **Field::Element_ptr** X, const size_t incX)
use cblas_(d)scal when possible
- Global **FFLAS::reduce** (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t n, FFLAS_ELT *X, const size_t incX)
use cblas_(d)scal when possible
- Global **FFLAS::reduce** (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t n, const FFLAS_ELT *Y, const size_t incY, FFLAS_ELT *X, const size_t incX)
use cblas_(d)scal when possible
- Global **FFLAS::fscal** (const Field &F, const size_t n, const typename **Field::Element** alpha, typename **Field::ConstElement_ptr** X, const size_t incX, typename **Field::Element_ptr** Y, const size_t incY)
use cblas_(d)scal when possible
- Global **FFLAS::fscal** (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t n, const FFLAS_ELT alpha, const FFLAS_ELT *X, const size_t incX, FFLAS_ELT *Y, const size_t incY)
use cblas_(d)scal when possible
- Global **FFLAS::fscaln** (const Field &F, const size_t n, const typename **Field::Element** alpha, typename **Field::Element_ptr** X, const size_t incX)
use cblas_(d)scal when possible
- Global **FFLAS::fscaln** (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t n, const FFLAS_ELT alpha, FFLAS_ELT *X, const size_t incX)
use cblas_(d)scal when possible
- Global **FFLAS::fsquare** (const Field &F, const FFLAS_TRANSPOSE ta, const size_t n, const typename **Field::Element** alpha, typename **Field::ConstElement_ptr** A, const size_t lda, const typename **Field::Element** beta, typename **Field::Element_ptr** C, const size_t ldc)
why double ?
- Global **FFLAS::fswap** (const Field &F, const size_t N, typename **Field::Element_ptr** X, const size_t incX, typename **Field::Element_ptr** Y, const size_t incY)
use cblas_dswap when double
- Global **FFLAS::fswap** (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t N, FFLAS_ELT *X, const size_t incX, FFLAS_ELT *Y, const size_t incY)
use cblas_dswap when double
- Global **FFLAS::ftrsm** (const Field &F, const FFLAS_SIDE Side, const FFLAS_UPLO Uplo, const FFLAS_TRANSPOSE TransA, const FFLAS_DIAG Diag, const size_t M, const size_t N, const typename **Field::Element** alpha, typename **Field::ConstElement_ptr** A, const size_t lda, typename **Field::Element_ptr** B, const size_t ldb)
 α must be non zero.
- Global **FFLAS::ftrsm** (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS_SIDE Side, const FFLAS_UPLO Uplo, const FFLAS_TRANSPOSE TransA, const FFLAS_DIAG Diag, const size_t M, const size_t N, const FFLAS_ELT alpha, const FFLAS_ELT *A, const size_t lda, FFLAS_ELT *B, const size_t ldb)
 α must be non zero.
- Global **FFPACK::buildMatrix** (const Field &F, typename **Field::ConstElement_ptr** E, typename **Field::ConstElement_ptr** C, const size_t lda, const size_t *B, const size_t *T, const size_t me, const size_t mc, const size_t lambda, const size_t mu)
is this :
- Global **FFPACK::invert2** (const Field &F, const size_t M, typename **Field::Element_ptr** A, const size_t lda, typename **Field::Element_ptr** X, const size_t idx, int &>nullity)
not tested.
- Global **launch_fger_dispatch** (const Field &F, const size_t nn, const typename **Field::Element** alpha, const size_t iters, RandIter &G)
test for transpo
test for incx equal

Global `launch_MM_dispatch` (const Field &F, const int mm, const int nn, const int kk, const typename `Field::Element` alpha, const typename `Field::Element` beta, const size_t iters, RandIter &G)

Global `launch_MM_dispatch` (const Field &F, const int mm, const int nn, const int kk, const typename `Field::Element` alpha, const typename `Field::Element` beta, const size_t iters, const int nbw, const bool par, RandIter &G)

test for IdX equal

test for transpo

Global `launch_MM_dispatch` (const Field &F, const int mm, const int nn, const int kk, const typename `Field::Element` alpha, const typename `Field::Element` beta, const size_t iters, RandIter &G)

test for transpo

test for IdX equal

Global `printvect` (std::ostream &o, vector< T > &vect)

does not belong here

Chapter 3

Bibliography

- Global **FFLAS::Protected::TRSMBound** (const Givaro::ModularBalanced< Element > &F) .
Dumas Giorgi Pernet 06, arXiv:cs/0601133
- Global **FFPACK::LeadingSubmatrixRankProfiles** (const size_t M, const size_t N, const size_t R, const size_t LSm, const size_t LSn, const size_t *P, const size_t *Q, size_t *RRP, size_t *CRP) .
Dumas J-G., Pernet C., and Sultan Z. *Simultaneous computation of the row and column rank profiles*, ISSAC'13.
- Global **FFPACK::LUdivine** (const Field &F, const **FFLAS::FFLAS_DIAG** Diag, const **FFLAS::FFLAS_TRANSPOSE** trans, const size_t M, const size_t N, typename **Field::Element_ptr** A, const size_t lda, size_t *P, size_t *Qt, const FFPACK_LU_TAG LuTag=FfpackSlabRecursive, const size_t cutoff=__FFLASFFPACK_LUDIVINE_THRESHOLD) .
Jeannerod C-P, Pernet, C. and Storjohann, A. *Rank-profile revealing Gaussian elimination and the CUP matrix decomposition*, J. of Symbolic Comp., 2013
• Pernet C, Brassel M *LUdivine, une divine factorisation LU*, 2002
- Global **FFPACK::PLUQ** (const Field &F, const **FFLAS::FFLAS_DIAG** Diag, const size_t M, const size_t N, typename **Field::Element_ptr** A, const size_t lda, size_t *P, size_t *Q) .
Dumas J-G., Pernet C., and Sultan Z. *Simultaneous computation of the row and column rank profiles*, ISSAC'13, 2013
- Global **FFPACK::productBruhatxTS** (const Field &Fi, size_t N, size_t s, size_t r, size_t t, const size_t *P, const size_t *Q, typename **Field::ConstElement_ptr** Xu, size_t ldu, size_t NbBlocksU, const size_t *Ku, const size_t *Tu, const size_t *MU, typename **Field::ConstElement_ptr** XI, size_t ldl, size_t NbBlocksL, const size_t *KI, const size_t *TI, const size_t *ML, typename **Field::Element_ptr** B, size_t ldb, const typename **Field::Element** beta, typename **Field::Element_ptr** D, size_t ldd) .
Pernet C. and Storjohann A. *Time and space efficient generators for quasiseparable matrices*, JSC (85), 2018, doi:10.1016/j.jsc.2017.07.010
- Global **FFPACK::Protected::GaussJordan** (const Field &F, const size_t M, const size_t N, typename **Field::Element_ptr** A, const size_t lda, const size_t colbeg, const size_t rowbeg, const size_t colsize, size_t *P, size_t *Q, const FFPACK::FFPACK_LU_TAG LuTag) .
Algorithm 2.8 of A. Storjohann Thesis 2000,
• Algorithm 11 of Jeannerod C-P, Pernet, C. and Storjohann, A. *Rank-profile revealing Gaussian elimination and the CUP matrix decomposition*, J. of Symbolic Comp., 2013
- Class **ftrsmLeftUpperNoTransNonUnit**< Element > .
Dumas, Giorgi, Pernet 06, arXiv:cs/0601133.

Chapter 4

Todo List

File `debug.h`

we should put vector printing elsewhere.

Global `FFLAS::fadd` (const Field &F, const size_t N, typename `Field::ConstElement_ptr` A, const size_t incA, const typename `Field::Element` alpha, typename `Field::ConstElement_ptr` B, const size_t incB, typename `Field::Element_ptr` C, const size_t incC)

optimise here

Global `FFLAS::fassign` (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t N, const FFLAS_ELT *Y, const size_t incY, FFLAS_ELT *X, const size_t incX)

variant for triangular matrix

Global `FFLAS::fassign` (const Field &F, const size_t N, typename `Field::ConstElement_ptr` Y, const size_t incY, typename `Field::Element_ptr` X, const size_t incX)

variant for triangular matrix

Global `FFLAS::fconvert` (const Field &F, const size_t m, const size_t n, OtherElement_ptr A, const size_t lda, typename `Field::ConstElement_ptr` B, const size_t ldb)

check if n == lda

Global `FFLAS::fneg` (const Field &F, const size_t m, const size_t n, typename `Field::ConstElement_ptr` B, const size_t ldb, typename `Field::Element_ptr` A, const size_t lda)

check if n == lda

Global `FFLAS::fnegin` (const Field &F, const size_t m, const size_t n, typename `Field::Element_ptr` A, const size_t lda)

check if n == lda

Global `FFLAS::fscal` (const Field &F, const size_t n, const typename `Field::Element` alpha, typename `Field::ConstElement_ptr` X, const size_t incX, typename `Field::Element_ptr` Y, const size_t incY)

check if comparison with +/-1,0 is necessary.

Global `FFLAS::fscal` (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t n, const FFLAS_ELT alpha, const FFLAS_ELT *X, const size_t incX, FFLAS_ELT *Y, const size_t incY)

check if comparison with +/-1,0 is necessary.

Global `FFLAS::fscaln` (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t n, const FFLAS_ELT alpha, FFLAS_ELT *X, const size_t incX)

check if comparison with +/-1,0 is necessary.

Global `FFLAS::fscaln` (const Field &F, const size_t n, const typename `Field::Element` alpha, typename `Field::Element_ptr` X, const size_t incX)

check if comparison with +/-1,0 is necessary.

Global **FFLAS::Protected::igemm** (const enum FFLAS_TRANSPOSE TransA, const enum FFLAS_TRANSPOSE TransB, size_t rows, size_t cols, size_t depth, const int64_t alpha, const int64_t *A, size_t lda, const int64_t *B, size_t ldb, const int64_t beta, int64_t *C, size_t ldc)
use primitive (no **Field()**) and specialise for int64.

Global **FFLAS::Protected::MatF2MatFI_Triangular** (const **Field** &F, **Givaro::FloatDomain::Element_ptr** S, const size_t lds, typename **Field::ConstElement_ptr** const E, const size_t lde, const size_t m, const size_t n)
do finit(...,FFLAS_TRANS,FFLAS_DIAG)
do fconvert(...,FFLAS_TRANS,FFLAS_DIAG)

Global **FFPACK::getTriangular** (const **Field** &F, const **FFLAS::FFLAS_UPLO** Uplo, const **FFLAS::FFLAS_DIAG** diag, const size_t M, const size_t N, const size_t R, typename **Field::ConstElement_ptr** A, const size_t lda, typename **Field::Element_ptr** T, const size_t ldt, const bool OnlyNonZeroVectors=false)
just one triangular fzero+fassign ?

Global **FFPACK::getTriangular** (const **Field** &F, const **FFLAS::FFLAS_UPLO** Uplo, const **FFLAS::FFLAS_DIAG** diag, const size_t M, const size_t N, const size_t R, typename **Field::Element_ptr** A, const size_t lda)
just one triangular fzero+fassign ?

Global **FFPACK::invert2** (const **Field** &F, const size_t M, typename **Field::Element_ptr** A, const size_t lda, typename **Field::Element_ptr** X, const size_t ldx, int &>nullity)
this init is not all necessary (done after ftrtri)

Global **FFPACK::LUdivine** (const **Field** &F, const **FFLAS::FFLAS_DIAG** Diag, const **FFLAS::FFLAS_TRANSPOSE** trans, const size_t M, const size_t N, typename **Field::Element_ptr** A, const size_t lda, size_t *P, size_t *Q, const **FFPACK::FFPACK_LU_TAG** LuTag, const size_t cutoff)
std::swap ?

Global **FFPACK::Protected::RandomKrylovPrecond** (const **PolRing** &PR, std::list< typename **PolRing::Element** > &completedFactors, const size_t N, typename **PolRing::Domain_t::Element_ptr** A, const size_t lda, size_t &Nb, typename **PolRing::Domain_t::Element_ptr** &B, size_t &ldb, typename **PolRing::Domain_t::RandIter** &g, const size_t degree=__FFLASFFPACK_ARITHPROG_THRESHOLD)
don't assing K2 c*noc x N but only mas (c,noc) x N and store each one after the other

swap to save space ??

Module **field**

biblio

Global **launch_fger_dispatch** (const **Field** &F, const size_t nn, const typename **Field::Element** alpha, const size_t iters, **RandIter** &G)
does nbw actually do nbw recursive calls and then call blas (check ?) ?

Global **launch_MM_dispatch** (const **Field** &F, const int mm, const int nn, const int kk, const typename **Field::Element** alpha, const typename **Field::Element** beta, const size_t iters, **RandIter** &G)
does nbw actually do nbw recursive calls and then call blas (check ?) ?

Global **launch_MM_dispatch** (const **Field** &F, const int mm, const int nn, const int kk, const typename **Field::Element** alpha, const typename **Field::Element** beta, const size_t iters, const int nbw, const bool par, **RandIter** &G)
does nbw actually do nbw recursive calls and then call blas (check ?) ?

Module **MMalgos**

biblio

Module **simd**

biblio

Global **test_colechelon** (**Field** &F, size_t m, size_t n, size_t r, size_t iters, **FFPACK::FFPACK_LU_TAG** LuTag, **RandIter** &G, bool par)
check lda

Global **test_det** (Field &F, size_t n, int iter, RandIter &G)

test with stride

Global **test_redcochelon** (Field &F, size_t m, size_t n, size_t r, size_t iters, FFPACK::FFPACK_LU_TAG LuTag, RandIter &G, bool par)

check Ida

Global **test_redrowechelon** (Field &F, size_t m, size_t n, size_t r, size_t iters, FFPACK::FFPACK_LU_TAG LuTag, RandIter &G, bool par)

check Ida

Global **test_rowechelon** (Field &F, size_t m, size_t n, size_t r, size_t iters, FFPACK::FFPACK_LU_TAG LuTag, RandIter &G, bool par)

check Ida

Chapter 5

Topic Index

5.1 Topics

Here is a list of all topics with brief descriptions:

CHECKER	41
FFLAS-FFPACK	41
FFLAS	42
Interfaces	42
Matrix Multiplication Algorithms	42
SIMD wrapper	42
FFPACK	43
FFLAS-FFPACK fields	43
RNS	43

Chapter 6

Namespace Index

6.1 Namespace List

Here is a list of all namespaces with brief descriptions:

FFLAS	45
FFLAS::_ftranspose_impl	194
FFLAS::BLAS3	195
FFLAS::csr_hyb_details	201
FFLAS::CuttingStrategy	201
FFLAS::details	202
FFLAS::details_spmv	210
FFLAS::ElementCategories	211
FFLAS::FieldCategories	
Traits and categories will need to be placed in a proper file later	211
FFLAS::MMHelperAlgo	211
FFLAS::ModeCategories	
Specifies the mode of action for an algorithm w.r.t	212
FFLAS::ParSeqHelper	
ParSeqHelper for both fgemm and ftrsm	212
FFLAS::Protected	213
FFLAS::sell_details	227
FFLAS::sparse_details	228
FFLAS::sparse_details_impl	242
FFLAS::StrategyParameter	282
FFLAS::StructureHelper	
StructureHelper for ftrsm	282
FFLAS::vectorised	283
FFLAS::vectorised::unswitch	288
FFPACK	
Finite Field PACK Set of elimination based routines for dense linear algebra	291
FFPACK::Protected	396
Givaro	403
MKL_CONFIG	403
RecInt	403

Chapter 7

Hierarchical Index

7.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

AlgoChooser< ModeT, ParSeq >	405
AlgoChooser< ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeq >	405
ALL< v >	405
ALL< false, v... >	406
ALL< true, v... >	406
ALL<>	406
ArbitraryPrecIntTag	406
AreEqual< X, Y >	407
AreEqual< X, X >	407
Argument	407
associatedDelayedField< Field >	408
associatedDelayedField< const FFPACK::RNSIntegerMod< RNS > >	408
associatedDelayedField< const Givaro::Modular< T, X > >	409
associatedDelayedField< const Givaro::ModularBalanced< T > >	409
associatedDelayedField< const Givaro::ZRing< T > >	410
Auto	410
Bench< Elt >	410
Bini	413
Block	413
BlockTransposeSIMD< Field, Simd, >	413
callLUdivine_small< Element >	415
callLUdivine_small< double >	415
callLUdivine_small< float >	416
CharpolyFailed	416
Checker_Empty< Field >	416
CheckerImplem_charpoly< Field, Polynomial >	417
CheckerImplem_charpoly< Givaro::ZRing< Givaro::Integer >, Polynomial >	418
CheckerImplem_Det< Field >	419
CheckerImplem_fgemm< Field >	421
CheckerImplem_ftrsm< Field >	422
CheckerImplem_invert< Field >	423
CheckerImplem_PLUQ< Field >	424
Classic	425
Column	425
CompactElement< Element >	425

CompactElement< double >	425
CompactElement< float >	426
CompactElement< int16_t >	426
CompactElement< int32_t >	426
CompactElement< int64_t >	426
compatible_data_type< Field >	427
compatible_data_type< Givaro::ZRing< double > >	427
compatible_data_type< Givaro::ZRing< float > >	427
Compose< H1, H2 >	427
Simd128_impl< true, true, false, 2 >::Converter	429
Simd128_impl< true, true, false, 4 >::Converter	429
Simd128_impl< true, true, false, 8 >::Converter	429
Simd128_impl< true, true, true, 2 >::Converter	430
Simd128_impl< true, true, true, 4 >::Converter	430
Simd128_impl< true, true, true, 8 >::Converter	430
Simd256_impl< true, false, true, 8 >::Converter	431
Simd256_impl< true, true, false, 2 >::Converter	431
Simd256_impl< true, true, false, 4 >::Converter	431
Simd256_impl< true, true, false, 8 >::Converter	432
Simd256_impl< true, true, true, 2 >::Converter	432
Simd256_impl< true, true, true, 4 >::Converter	432
Simd256_impl< true, true, true, 8 >::Converter	433
Simd512_impl< true, true, false, 8 >::Converter	433
Simd512_impl< true, true, true, 8 >::Converter	433
ConvertTo< T >	434
Coo< ValT, IdxT >	434
Coo< Field >	435
Coo< ValT, IdxT >	437
CooMat< Field >	438
CooMat< FFPACK::RNSInteger >	438
count_nonconst_lvalue_reference< T >	439
count_nonconst_lvalue_reference< const T &, O... >	439
count_nonconst_lvalue_reference< T &, O... >	440
count_nonconst_lvalue_reference< T, O... >	440
count_nonconst_lvalue_reference<>	440
CsrMat< Field >	440
CsrMat< FFPACK::RNSInteger >	440
DefaultBoundedTag	441
DefaultTag	441
DelayedTag	442
DivideAndConquer	442
ElementTraits< Element >	442
ElementTraits< double >	442
ElementTraits< FFPACK::rns_double_elt >	443
ElementTraits< float >	443
ElementTraits< Givaro::Integer >	443
ElementTraits< int16_t >	444
ElementTraits< int32_t >	444
ElementTraits< int64_t >	444
ElementTraits< int8_t >	444
ElementTraits< Reclnt::rint< K > >	445
ElementTraits< Reclnt::rmint< K, MG > >	445
ElementTraits< Reclnt::ruint< K > >	445
ElementTraits< uint16_t >	446
ElementTraits< uint32_t >	446
ElementTraits< uint64_t >	446
ElementTraits< uint8_t >	447
ElMat< Field >	447

EllMat< FFPACK::RNSInteger >	447
Failure	448
FailureCharpolyCheck	449
FailureDetCheck	449
FailureFgemmCheck	449
FailureInvertCheck	450
FailurePLUQCheck	450
FailureTrsmCheck	450
false_type	
isSparseMatrix< Field, M >	479
isSparseMatrixMKLFormat< F, M >	483
isSparseMatrixSimdFormat< F, M >	483
isZOSparseMatrix< F, M >	483
support_fast_mod< T >	757
support_simd< T >	758
support_simd_add< T >	759
support_simd_mod< T >	759
FieldSimd< _Field >	450
FieldTraits< Field >	456
FieldTraits< FFPACK::RNSInteger< T > >	457
FieldTraits< FFPACK::RNSIntegerMod< T > >	457
FieldTraits< Givaro::Modular< Element > >	458
FieldTraits< Givaro::ModularBalanced< Element > >	458
FieldTraits< Givaro::ZRing< double > >	459
FieldTraits< Givaro::ZRing< float > >	459
FieldTraits< Givaro::ZRing< Givaro::Integer > >	460
FieldTraits< Givaro::ZRing< int16_t > >	460
FieldTraits< Givaro::ZRing< int32_t > >	461
FieldTraits< Givaro::ZRing< int64_t > >	461
FieldTraits< Givaro::ZRing< Reclnt::ruint< K > > >	462
FieldTraits< Givaro::ZRing< uint16_t > >	462
FieldTraits< Givaro::ZRing< uint32_t > >	462
FieldTraits< Givaro::ZRing< uint64_t > >	463
Fixed	463
FixedPreclntTag	463
ScalFunctionsBase< Element, typename enable_if< is_floating_point< Element >::value >::type >← ::FloatingPointTestDistribution	464
ftmmLeftLowerNoTransNonUnit< Element >	464
ftmmLeftLowerNoTransUnit< Element >	464
ftmmLeftLowerTransNonUnit< Element >	465
ftmmLeftLowerTransUnit< Element >	465
ftmmLeftUpperNoTransNonUnit< Element >	465
ftmmLeftUpperNoTransUnit< Element >	465
ftmmLeftUpperTransNonUnit< Element >	465
ftmmLeftUpperTransUnit< Element >	465
ftmmRightLowerNoTransNonUnit< Element >	465
ftmmRightLowerNoTransUnit< Element >	465
ftmmRightLowerTransNonUnit< Element >	466
ftmmRightLowerTransUnit< Element >	466
ftmmRightUpperNoTransNonUnit< Element >	466
ftmmRightUpperNoTransUnit< Element >	466
ftmmRightUpperTransNonUnit< Element >	466
ftmmRightUpperTransUnit< Element >	466
ftsmLeftLowerNoTransNonUnit< Element >	466
ftsmLeftLowerNoTransUnit< Element >	466
ftsmLeftLowerTransNonUnit< Element >	467
ftsmLeftLowerTransUnit< Element >	467
ftsmLeftUpperNoTransNonUnit< Element >	467

ftsmLeftUpperNoTransUnit< Element >	467
ftsmLeftUpperTransNonUnit< Element >	467
ftsmLeftUpperTransUnit< Element >	467
ftsmRightLowerNoTransNonUnit< Element >	468
ftsmRightLowerNoTransUnit< Element >	468
ftsmRightLowerTransNonUnit< Element >	468
ftsmRightLowerTransUnit< Element >	468
ftsmRightUpperNoTransNonUnit< Element >	468
ftsmRightUpperNoTransUnit< Element >	468
ftsmRightUpperTransNonUnit< Element >	468
ftsmRightUpperTransUnit< Element >	468
GenericTag	469
GenericTag	469
Grain	469
has_minus_eq_impl< C >	469
has_minus_impl< C >	469
has_mul_eq_impl< C >	470
has_mul_impl< C >	470
has_operation< T >	470
has_plus_eq_impl< C >	471
has_plus_impl< C >	471
HelperFlag	471
HelperMod< Field, ElementTraits >	472
HelperMod< Field, ElementCategories::MachineIntTag >	472
HelperMod< Field, FFLAS::ElementCategories::ArbitraryPrecIntTag >	473
HelperMod< Field, FFLAS::ElementCategories::FixedPrecIntTag >	474
HelperMod< Field, FFLAS::ElementCategories::MachineFloatTag >	474
Hybrid	475
Info	475
Info	477
is_all_same< Args >	478
is_all_same< T, Args... >	478
is_all_same<>	478
is_simd< T >	478
Iterative	485
LazyTag	485
limits< T >	485
limits< char >	485
limits< double >	486
limits< float >	487
limits< Givaro::Integer >	487
limits< int >	488
limits< long >	488
limits< long long >	489
limits< ReclInt::rint< K > >	490
limits< ReclInt::ruint< K > >	490
limits< short int >	491
limits< signed char >	492
limits< unsigned char >	492
limits< unsigned int >	493
limits< unsigned long >	493
limits< unsigned long long >	494
limits< unsigned short int >	495
MachineFloatTag	495
MachineIntTag	495
MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait >	496
MMHelper< FFPACK::RNSInteger< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >	501
MMHelper< FFPACK::RNSIntegerMod< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >	503

MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< Dest >, ParSeqTrait >	505
MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeqTrait >	506
MMHelper< Field, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >	508
ModeTraits< Field >	510
ModeTraits< Givaro::Modular< Element, Compute > >	510
ModeTraits< Givaro::Modular< Givaro::Integer, Compute > >	511
ModeTraits< Givaro::Modular< int16_t, Compute > >	511
ModeTraits< Givaro::Modular< int32_t, Compute > >	511
ModeTraits< Givaro::Modular< int64_t, uint64_t > >	512
ModeTraits< Givaro::Modular< int8_t, Compute > >	512
ModeTraits< Givaro::Modular< Reclnt::ruint< K >, Compute > >	512
ModeTraits< Givaro::Modular< uint16_t, Compute > >	513
ModeTraits< Givaro::Modular< uint32_t, Compute > >	513
ModeTraits< Givaro::Modular< uint8_t, Compute > >	513
ModeTraits< Givaro::ModularBalanced< Element > >	514
ModeTraits< Givaro::ModularBalanced< Givaro::Integer > >	514
ModeTraits< Givaro::ModularBalanced< int16_t > >	514
ModeTraits< Givaro::ModularBalanced< int32_t > >	515
ModeTraits< Givaro::ModularBalanced< int8_t > >	515
ModeTraits< Givaro::Montgomery< T > >	515
ModeTraits< Givaro::ZRing< double > >	515
ModeTraits< Givaro::ZRing< float > >	516
ModeTraits< Givaro::ZRing< Givaro::Integer > >	516
ModularBalanced< T >	516
ModularTag	516
Montgomery< T >	517
need_field_characteristic< Field >	517
need_field_characteristic< Givaro::Modular< Field > >	517
need_field_characteristic< Givaro::ModularBalanced< Field > >	517
NoSimd< T >	518
Parallel< C, P >	519
readMyMachineType< Field, T >	522
readMyMachineType< Field, mpz_t >	523
Recursive	524
Recursive	524
rint< K >	524
rns_double	524
rns_double_elt	529
rns_double_elt_cstptr	530
rns_double_elt_ptr	533
rns_double_extended	536
RNSElementTag	539
RNSInteger< RNS >	539
RNSInteger< FFPACK::rns_double >	539
RNSIntegerMod< RNS >	543
RNSIntegerMod< FFPACK::rns_double >	543
rnsRandIter< RNS >	550
RNSInteger< RNS >::RandIter	520
RNSIntegerMod< RNS >::RandIter	521
Row	551
ruint< K >	551
ScalFunctionsBase< Element, Enable >	556
ScalFunctions< Element >	551
ScalFunctionsBase< Element, typename enable_if< is_floating_point< Element >::value >::type >	557
ScalFunctionsBase< Element, typename enable_if< is_integral< Element >::value >::type >	558
Sequential	561

Simd128_impl< ArithType, Int, Signed, Size >	562
Simd128_impl< true, false, true, 4 >	562
Simd128_impl< true, false, true, 8 >	562
Simd128i_base	620
Simd128_impl< true, true, true, 2 >	592
Simd128_impl< true, true, false, 2 >	562
Simd128_impl< true, true, true, 4 >	602
Simd128_impl< true, true, false, 4 >	572
Simd128_impl< true, true, true, 8 >	610
Simd128_impl< true, true, false, 8 >	582
Simd256_impl< ArithType, Int, Signed, Size >	621
Simd256fp_base	703
Simd256_impl< true, false, true, 4 >	621
Simd256_impl< true, false, true, 8 >	622
Simd256i_base	703
Simd256_impl< true, true, true, 2 >	667
Simd256_impl< true, true, false, 2 >	629
Simd256_impl< true, true, true, 4 >	676
Simd256_impl< true, true, false, 4 >	639
Simd256_impl< true, true, false, 4 >	639
Simd256_impl< true, true, true, 8 >	694
Simd256_impl< true, true, false, 8 >	658
Simd512_impl< ArithType, Int, Signed, Size >	703
Simd512_impl< true, false, true, 4 >	703
Simd512_impl< true, false, true, 8 >	704
Simd512i_base	731
Simd256_impl< true, true, true, 4 >	676
Simd512_impl< true, true, true, 8 >	721
Simd512_impl< true, true, false, 8 >	710
SimdChooser< T, bool, bool >	732
SimdChooser< T, false, b >	732
SimdChooser< T, true, false >	732
SimdChooser< T, true, true >	733
simdToType< T >	733
Single	733
Sparse< Field, SparseMatrix_t, IdxT, PtrT >	733
Sparse< _Field, SparseMatrix_t::COO >	733
Sparse< _Field, SparseMatrix_t::COO_ZO >	735
Sparse< _Field, SparseMatrix_t::CSR >	737
Sparse< _Field, SparseMatrix_t::CSR_ZO >	740
Sparse< _Field, SparseMatrix_t::CSR_HYB >	738
Sparse< _Field, SparseMatrix_t::ELL >	742
Sparse< _Field, SparseMatrix_t::ELL_ZO >	747
Sparse< _Field, SparseMatrix_t::ELL_simd >	743
Sparse< _Field, SparseMatrix_t::ELL_simd_ZO >	745
Sparse< _Field, SparseMatrix_t::HYB_ZO >	748
Sparse< _Field, SparseMatrix_t::SELL >	750
Sparse< _Field, SparseMatrix_t::SELL_ZO >	752
Sparse< FFPACK::RNSInteger, SparseMatrix_t::COO, int16_t >	733
Sparse< FFPACK::RNSInteger, SparseMatrix_t::COO, int32_t >	733
Sparse< FFPACK::RNSInteger, SparseMatrix_t::COO, int64_t >	733
Sparse< FFPACK::RNSInteger, SparseMatrix_t::COO_ZO, int16_t >	733
Sparse< FFPACK::RNSInteger, SparseMatrix_t::COO_ZO, int32_t >	733
Sparse< FFPACK::RNSInteger, SparseMatrix_t::COO_ZO, int64_t >	733
Sparse< FFPACK::RNSInteger, SparseMatrix_t::CSR, int16_t >	733

Sparse< FFPACK::RNSInteger, SparseMatrix_t::CSR, int32_t >	733
Sparse< FFPACK::RNSInteger, SparseMatrix_t::CSR, int64_t >	733
Sparse< FFPACK::RNSInteger, SparseMatrix_t::CSR_ZO, int16_t >	733
Sparse< FFPACK::RNSInteger, SparseMatrix_t::CSR_ZO, int32_t >	733
Sparse< FFPACK::RNSInteger, SparseMatrix_t::CSR_ZO, int64_t >	733
Sparse< FFPACK::RNSInteger, SparseMatrix_t::ELL, int16_t >	733
Sparse< FFPACK::RNSInteger, SparseMatrix_t::ELL, int32_t >	733
Sparse< FFPACK::RNSInteger, SparseMatrix_t::ELL, int64_t >	733
Sparse< FFPACK::RNSInteger, SparseMatrix_t::ELL_ZO, int16_t >	733
Sparse< FFPACK::RNSInteger, SparseMatrix_t::ELL_ZO, int32_t >	733
Sparse< FFPACK::RNSInteger, SparseMatrix_t::ELL_ZO, int64_t >	733
SpMat< Field, flag >	754
StatsMatrix	754
Test< Elt >	759
TestOneMethod< Simd >	762
tfn_minus	765
tfn_minus_eq	765
tfn_mul	766
tfn_mul_eq	766
tfn_plus	766
tfn_plus_eq	767
Threads	767
ThreeD	767
ThreeDAdaptive	767
ThreeDInPlace	768
TRSMHelper< ReclerTrait, ParSeqTrait >	768
true_type	
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::COO > >	479
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::COO_ZO > >	479
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR > >	480
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR_HYB > >	480
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR_ZO > >	480
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL > >	481
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_ZO > >	482
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_simd > >	481
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_simd_ZO > >	481
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::HYB_ZO > >	482
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::SELL > >	482
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::SELL_ZO > >	482
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::COO_ZO > >	484
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR_ZO > >	484
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_ZO > >	484
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_simd_ZO > >	484
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::SELL_ZO > >	485
support_fast_mod< double >	757
support_fast_mod< float >	758
support_fast_mod< int64_t >	758
TwoD	769
TwoDAdaptive	769
UnparametricTag	769
width< T >	770
width< double >	770
width< float >	770
Winograd	770
WinogradPar	770

Chapter 8

Data Structure Index

8.1 Data Structures

Here are the data structures with brief descriptions:

AlgoChooser< ModeT, ParSeq >	405
AlgoChooser< ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeq >	405
ALL< v >	405
ALL< false, v... >	406
ALL< true, v... >	406
ALL<>	406
ArbitraryPrecIntTag	
Arbitrary precision integers: GMP	406
AreEqual< X, Y >	407
AreEqual< X, X >	407
Argument	407
associatedDelayedField< Field >	408
associatedDelayedField< const FFPACK::RNSIntegerMod< RNS > >	408
associatedDelayedField< const Givaro::Modular< T, X > >	409
associatedDelayedField< const Givaro::ModularBalanced< T > >	409
associatedDelayedField< const Givaro::ZRing< T > >	410
Auto	410
Bench< Elt >	410
Bini	413
Block	413
BlockTransposeSIMD< Field, Simd, >	413
callLUdivine_small< Element >	415
callLUdivine_small< double >	415
callLUdivine_small< float >	416
CharpolyFailed	416
Checker_Empty< Field >	416
CheckerImplem_charpoly< Field, Polynomial >	417
CheckerImplem_charpoly< Givaro::ZRing< Givaro::Integer >, Polynomial >	418
CheckerImplem_Det< Field >	419
CheckerImplem_fgemm< Field >	421
CheckerImplem_ftsrm< Field >	422
CheckerImplem_invert< Field >	423
CheckerImplem_PLUQ< Field >	424
Classic	425
Column	425

CompactElement< Element >	425
CompactElement< double >	425
CompactElement< float >	426
CompactElement< int16_t >	426
CompactElement< int32_t >	426
CompactElement< int64_t >	426
compatible_data_type< Field >	427
compatible_data_type< Givaro::ZRing< double > >	427
compatible_data_type< Givaro::ZRing< float > >	427
Compose< H1, H2 >	427
Simd128_impl< true, true, false, 2 >::Converter	429
Simd128_impl< true, true, false, 4 >::Converter	429
Simd128_impl< true, true, false, 8 >::Converter	429
Simd128_impl< true, true, true, 2 >::Converter	430
Simd128_impl< true, true, true, 4 >::Converter	430
Simd128_impl< true, true, true, 8 >::Converter	430
Simd256_impl< true, false, true, 8 >::Converter	431
Simd256_impl< true, true, false, 2 >::Converter	431
Simd256_impl< true, true, false, 4 >::Converter	431
Simd256_impl< true, true, false, 8 >::Converter	432
Simd256_impl< true, true, true, 2 >::Converter	432
Simd256_impl< true, true, true, 4 >::Converter	432
Simd256_impl< true, true, true, 8 >::Converter	433
Simd512_impl< true, true, false, 8 >::Converter	433
Simd512_impl< true, true, true, 8 >::Converter	433
ConvertTo< T >	
Force conversion to appropriate element type of ElementCategory T	434
Coo< ValT, IdxT >	434
Coo< Field >	435
Coo< ValT, IdxT >	437
CooMat< Field >	438
count_nonconst_lvalue_reference< T >	439
count_nonconst_lvalue_reference< const T &, O... >	439
count_nonconst_lvalue_reference< T &, O... >	440
count_nonconst_lvalue_reference< T, O... >	440
count_nonconst_lvalue_reference<>	440
CsrMat< Field >	440
DefaultBoundedTag	
Use standard field operations, but keeps track of bounds on input and output	441
DefaultTag	
No specific mode of action: use standard field operations	441
DelayedTag	
Performs field operations with delayed mod reductions. Ensures result is reduced	442
DivideAndConquer	442
ElementTraits< Element >	
ElementTraits	442
ElementTraits< double >	442
ElementTraits< FFPACK::rns_double_elt >	443
ElementTraits< float >	443
ElementTraits< Givaro::Integer >	443
ElementTraits< int16_t >	444
ElementTraits< int32_t >	444
ElementTraits< int64_t >	444
ElementTraits< int8_t >	444
ElementTraits< Reclnt::rint< K > >	445
ElementTraits< Reclnt::rmint< K, MG > >	445
ElementTraits< Reclnt::ruint< K > >	445
ElementTraits< uint16_t >	446

ElementTraits< uint32_t >	446
ElementTraits< uint64_t >	446
ElementTraits< uint8_t >	447
ElMat< Field >	447
Failure	
A precondition failed	448
FailureCharpolyCheck	449
FailureDetCheck	449
FailureFgemmCheck	449
FailureInvertCheck	450
FailurePLUQCheck	450
FailureTrsmCheck	450
FieldSimd< _Field >	450
FieldTraits< Field >	
FieldTrait	456
FieldTraits< FFPACK::RNSInteger< T > >	457
FieldTraits< FFPACK::RNSIntegerMod< T > >	457
FieldTraits< Givaro::Modular< Element > >	458
FieldTraits< Givaro::ModularBalanced< Element > >	458
FieldTraits< Givaro::ZRing< double > >	459
FieldTraits< Givaro::ZRing< float > >	459
FieldTraits< Givaro::ZRing< Givaro::Integer > >	460
FieldTraits< Givaro::ZRing< int16_t > >	460
FieldTraits< Givaro::ZRing< int32_t > >	461
FieldTraits< Givaro::ZRing< int64_t > >	461
FieldTraits< Givaro::ZRing< Reclnt::ruint< K > > >	462
FieldTraits< Givaro::ZRing< uint16_t > >	462
FieldTraits< Givaro::ZRing< uint32_t > >	462
FieldTraits< Givaro::ZRing< uint64_t > >	463
Fixed	463
FixedPrecIntTag	
Fixed precision integers above machine precision: Givaro::reclnt	463
ScalFunctionsBase< Element, typename enable_if< is_floating_point< Element >::value >::type >::FloatingPointTestDistribution	
464	
ftmmLeftLowerNoTransNonUnit< Element >	464
ftmmLeftLowerNoTransUnit< Element >	464
ftmmLeftLowerTransNonUnit< Element >	465
ftmmLeftLowerTransUnit< Element >	465
ftmmLeftUpperNoTransNonUnit< Element >	465
ftmmLeftUpperNoTransUnit< Element >	465
ftmmLeftUpperTransNonUnit< Element >	465
ftmmLeftUpperTransUnit< Element >	465
ftmmRightLowerNoTransNonUnit< Element >	465
ftmmRightLowerNoTransUnit< Element >	465
ftmmRightLowerTransNonUnit< Element >	466
ftmmRightLowerTransUnit< Element >	466
ftmmRightUpperNoTransNonUnit< Element >	466
ftmmRightUpperNoTransUnit< Element >	466
ftmmRightUpperTransNonUnit< Element >	466
ftmmRightUpperTransUnit< Element >	466
ftsmLeftLowerNoTransNonUnit< Element >	466
ftsmLeftLowerNoTransUnit< Element >	466
ftsmLeftLowerTransNonUnit< Element >	467
ftsmLeftLowerTransUnit< Element >	467
ftsmLeftUpperNoTransNonUnit< Element >	
Computes the maximal size for delaying the modular reduction in a triangular system resolution	467
ftsmLeftUpperNoTransUnit< Element >	467
ftsmLeftUpperTransNonUnit< Element >	467

ftsmLeftUpperTransUnit< Element >	467
ftsmRightLowerNoTransNonUnit< Element >	468
ftsmRightLowerNoTransUnit< Element >	468
ftsmRightLowerTransNonUnit< Element >	468
ftsmRightLowerTransUnit< Element >	468
ftsmRightUpperNoTransNonUnit< Element >	468
ftsmRightUpperNoTransUnit< Element >	468
ftsmRightUpperTransNonUnit< Element >	468
ftsmRightUpperTransUnit< Element >	468
GenericTag	
Default is generic	469
GenericTag	
Generic ring	469
Grain	469
has_minus_eq_impl< C >	469
has_minus_impl< C >	469
has_mul_eq_impl< C >	470
has_mul_impl< C >	470
has_operation< T >	470
has_plus_eq_impl< C >	471
has_plus_impl< C >	471
HelperFlag	471
HelperMod< Field, ElementTraits >	472
HelperMod< Field, ElementCategories::MachineIntTag >	472
HelperMod< Field, FFLAS::ElementCategories::ArbitraryPrecIntTag >	473
HelperMod< Field, FFLAS::ElementCategories::FixedPrecIntTag >	474
HelperMod< Field, FFLAS::ElementCategories::MachineFloatTag >	474
Hybrid	475
Info	475
Info	477
is_all_same< Args >	478
is_all_same< T, Args... >	478
is_all_same<>	478
is_simd< T >	478
isSparseMatrix< Field, M >	479
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::COO > >	479
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::COO_ZO > >	479
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR > >	480
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR_HYB > >	480
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR_ZO > >	480
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL > >	481
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_simd > >	481
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_simd_ZO > >	481
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_ZO > >	482
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::HYB_ZO > >	482
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::SELL > >	482
isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::SELL_ZO > >	482
isSparseMatrixMKLFormat< F, M >	483
isSparseMatrixSimdFormat< F, M >	483
isZOSparseMatrix< F, M >	483
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::COO_ZO > >	484
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR_ZO > >	484
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_simd_ZO > >	484
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_ZO > >	484
isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::SELL_ZO > >	485
Iterative	485
LazyTag	
Performs field operations with delayed mod only when necessary. Result may not be reduced	485

limits< T >	485
limits< char >	485
limits< double >	486
limits< float >	487
limits< Givaro::Integer >	487
limits< int >	488
limits< long >	488
limits< long long >	489
limits< RecInt::rint< K > >	490
limits< RecInt::ruint< K > >	490
limits< short int >	491
limits< signed char >	492
limits< unsigned char >	492
limits< unsigned int >	493
limits< unsigned long >	493
limits< unsigned long long >	494
limits< unsigned short int >	495
MachineFloatTag	
Float or double	495
MachineIntTag	
Short, int, long, long long, and unsigned variants	495
MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait >	496
MMHelper< FFPACK::RNSInteger< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >	501
MMHelper< FFPACK::RNSIntegerMod< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >	503
MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< Dest >, ParSeqTrait >	505
MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeqTrait >	506
MMHelper< Field, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >	
FGEMM Helper for Default and ConvertTo modes of operation	508
ModeTraits< Field >	
ModeTraits	510
ModeTraits< Givaro::Modular< Element, Compute > >	510
ModeTraits< Givaro::Modular< Givaro::Integer, Compute > >	511
ModeTraits< Givaro::Modular< int16_t, Compute > >	511
ModeTraits< Givaro::Modular< int32_t, Compute > >	511
ModeTraits< Givaro::Modular< int64_t, uint64_t > >	512
ModeTraits< Givaro::Modular< int8_t, Compute > >	512
ModeTraits< Givaro::Modular< RecInt::ruint< K >, Compute > >	512
ModeTraits< Givaro::Modular< uint16_t, Compute > >	513
ModeTraits< Givaro::Modular< uint32_t, Compute > >	513
ModeTraits< Givaro::Modular< uint8_t, Compute > >	513
ModeTraits< Givaro::ModularBalanced< Element > >	514
ModeTraits< Givaro::ModularBalanced< Givaro::Integer > >	514
ModeTraits< Givaro::ModularBalanced< int16_t > >	514
ModeTraits< Givaro::ModularBalanced< int32_t > >	515
ModeTraits< Givaro::ModularBalanced< int8_t > >	515
ModeTraits< Givaro::Montgomery< T > >	515
ModeTraits< Givaro::ZRing< double > >	515
ModeTraits< Givaro::ZRing< float > >	516
ModeTraits< Givaro::ZRing< Givaro::Integer > >	516
ModularBalanced< T >	516
ModularTag	
This is a modular field like e.g. Modular<T> or ModularBalanced<T>	516
Montgomery< T >	517
need_field_characteristic< Field >	517
need_field_characteristic< Givaro::Modular< Field > >	517
need_field_characteristic< Givaro::ModularBalanced< Field > >	517
NoSimd< T >	518

Parallel< C, P >	519
RNSInteger< RNS >::RandIter	520
RNSIntegerMod< RNS >::RandIter	521
readMyMachineType< Field, T >	522
readMyMachineType< Field, mpz_t >	523
Recursive	524
Recursive	524
rint< K >	524
rns_double	524
rns_double_elt	529
rns_double_elt_cstptr	530
rns_double_elt_ptr	533
rns_double_extended	536
RNSElementTag	
Representation in a Residue Number System	539
RNSInteger< RNS >	539
RNSIntegerMod< RNS >	543
rnsRandIter< RNS >	550
Row	551
ruint< K >	551
ScalFunctions< Element >	551
ScalFunctionsBase< Element, Enable >	556
ScalFunctionsBase< Element, typename enable_if< is_floating_point< Element >::value >::type >	557
ScalFunctionsBase< Element, typename enable_if< is_integral< Element >::value >::type >	558
Sequential	561
Simd128_impl< ArithType, Int, Signed, Size >	562
Simd128_impl< true, false, true, 4 >	562
Simd128_impl< true, false, true, 8 >	562
Simd128_impl< true, true, false, 2 >	562
Simd128_impl< true, true, false, 4 >	572
Simd128_impl< true, true, false, 8 >	582
Simd128_impl< true, true, true, 2 >	592
Simd128_impl< true, true, true, 4 >	602
Simd128_impl< true, true, true, 8 >	610
Simd128i_base	620
Simd256_impl< ArithType, Int, Signed, Size >	621
Simd256_impl< true, false, true, 4 >	621
Simd256_impl< true, false, true, 8 >	622
Simd256_impl< true, true, false, 2 >	629
Simd256_impl< true, true, false, 4 >	639
Simd256_impl< true, true, false, 8 >	658
Simd256_impl< true, true, true, 2 >	667
Simd256_impl< true, true, true, 4 >	676
Simd256_impl< true, true, true, 8 >	694
Simd256fp_base	703
Simd256i_base	703
Simd512_impl< ArithType, Int, Signed, Size >	703
Simd512_impl< true, false, true, 4 >	703
Simd512_impl< true, false, true, 8 >	704
Simd512_impl< true, true, false, 8 >	710
Simd512_impl< true, true, true, 8 >	721
Simd512i_base	731
SimdChooser< T, bool, bool >	732
SimdChooser< T, false, b >	732
SimdChooser< T, true, false >	732
SimdChooser< T, true, true >	733
simdToType< T >	733
Single	733

<code>Sparse< Field, SparseMatrix_t, IdxT, PtrT ></code>	733
<code>Sparse< _Field, SparseMatrix_t::COO ></code>	733
<code>Sparse< _Field, SparseMatrix_t::COO_ZO ></code>	735
<code>Sparse< _Field, SparseMatrix_t::CSR ></code>	737
<code>Sparse< _Field, SparseMatrix_t::CSR_HYB ></code>	738
<code>Sparse< _Field, SparseMatrix_t::CSR_ZO ></code>	740
<code>Sparse< _Field, SparseMatrix_t::ELL ></code>	742
<code>Sparse< _Field, SparseMatrix_t::ELL_simd ></code>	743
<code>Sparse< _Field, SparseMatrix_t::ELL_simd_ZO ></code>	745
<code>Sparse< _Field, SparseMatrix_t::ELL_ZO ></code>	747
<code>Sparse< _Field, SparseMatrix_t::HYB_ZO ></code>	748
<code>Sparse< _Field, SparseMatrix_t::SELL ></code>	750
<code>Sparse< _Field, SparseMatrix_t::SELL_ZO ></code>	752
<code>SpMat< Field, flag ></code>	754
<code>StatsMatrix</code>	754
<code>support_fast_mod< T ></code>	757
<code>support_fast_mod< double ></code>	757
<code>support_fast_mod< float ></code>	758
<code>support_fast_mod< int64_t ></code>	758
<code>support_simd< T ></code>	758
<code>support_simd_add< T ></code>	759
<code>support_simd_mod< T ></code>	759
<code>Test< Elt ></code>	759
<code>TestOneMethod< Simd ></code>	762
<code>tfn_minus</code>	765
<code>tfn_minus_eq</code>	765
<code>tfn_mul</code>	766
<code>tfn_mul_eq</code>	766
<code>tfn_plus</code>	766
<code>tfn_plus_eq</code>	767
<code>Threads</code>	767
<code>ThreeD</code>	767
<code>ThreeDAdaptive</code>	767
<code>ThreeDInPlace</code>	768
<code>TRSMHelper< ReclterTrait, ParSeqTrait ></code>	
<code>TRSM Helper</code>	768
<code>TwoD</code>	769
<code>TwoDAdaptive</code>	769
<code>UnparametricTag</code>	
If the field uses a representation with infix operators	769
<code>width< T ></code>	770
<code>width< double ></code>	770
<code>width< float ></code>	770
<code>Winograd</code>	770
<code>WinogradPar</code>	770

Chapter 9

File Index

9.1 File List

Here is a list of all files with brief descriptions:

arithprog.C	771
fsyrk.C	772
fsytrf.C	773
ftrtri.C	773
winograd.C	774
benchmark-charpoly-mp.C	775
benchmark-charpoly.C	776
benchmark-checkers.C	777
benchmark-dgemm.C	778
benchmark-dgetrf.C	779
benchmark-dgetri.C	779
benchmark-dsytrf.C	780
benchmark-dtrsm.C	781
benchmark-dtrtri.C	781
benchmark-fadd-lvl2.C	782
benchmark-fdot.C	783
benchmark-fgemm-mp.C	783
benchmark-fgemm-rns.C	784
benchmark-fgemm.C	786
benchmark-fgemv-mp.C	787
benchmark-fgemv.C	788
benchmark-fgesv.C	791
benchmark-fsyr2k.C	791
benchmark-fsyrk.C	792
benchmark-fsytrf.C	792
benchmark-ftrsm-mp.C	793
benchmark-ftrsm.C	794
benchmark-ftrsv.C	794
benchmark-ftrtri.C	795
benchmark-inverse.C	796
benchmark-lqup-mp.C	796
benchmark-lqup.C	797
benchmark-pluq.C	797
benchmark-quasisep.C	798
benchmark-storage-transpose.C	799

benchmark-wino.C	800
config.h	801
fflas-ffpack/config.h	804
mainpage.doxy	808
autotune/charpoly.C	808
examples/charpoly.C	808
det.C	809
matmul.C	809
autotune/pluq.C	810
examples/pluq.C	810
rank.C	811
solve.C	811
checker_charpoly.inl	812
checker_det.inl	812
checker_empty.h	812
checker_fgemm.inl	813
checker_ftrsm.inl	813
checker_invert.inl	813
checker_pluq.inl	814
checkers.doxy	814
checkers_fflas.h	814
checkers_fflas.inl	815
checkers_ffpack.h	815
checkers_ffpack.inl	816
config-blas.h	816
fflas-ffpack-config.h	
Defaults for optimised values	823
fflas-ffpack-default-thresholds.h	823
fflas-ffpack-thresholds.h	824
fflas-ffpack.doxy	824
fflas-ffpack.h	
Includes FFLAS and FFPACK	824
fflas.doxy	824
fflas.h	
Finite Field Linear Algebra Subroutines	824
fflas_bounds.inl	826
fflas_enum.h	827
fflas_fadd.h	827
fflas_fadd.inl	829
fflas_fassign.h	830
fflas_fassign.inl	830
fflas_faxpy.inl	831
fflas_fdot.inl	832
fflas_fgemm.inl	833
fgemm_classical.inl	835
fgemm_classical_mp.inl	
Matrix multiplication with multiprecision input (either over \mathbb{Z} or over $\mathbb{Z}/p\mathbb{Z}$)	835
fgemm_winograd.inl	837
matmul.doxy	838
schedule_bini.inl	
Bini implementation	838
schedule_winograd.inl	839
schedule_winograd_acc.inl	839
schedule_winograd_acc_ip.inl	840
schedule_winograd_ip.inl	841
fflas_fgemv.inl	841
fflas_fgemv_mp.inl	843
fflas_fger.inl	844

fflas_fger_mp.inl	845
fflas_freduce.h	846
fflas_freduce.inl	847
fflas_freduce_mp.inl	848
fflas_freivalds.inl	849
fflas_fscal.h	849
fflas_fscal.inl	849
fflas_fscal_mp.inl	851
fflas_fsyr2k.inl	851
fflas_fsyrk.inl	852
fflas_fsyrk_strassen.inl	854
fflas_ftmm.inl	855
fflas_ftsm.inl	855
fflas_ftsm_mp.inl	
Triangular system with matrix right hand side over multiprecision domain (either over \mathbb{Z} or over $\mathbb{Z}/p\mathbb{Z}$)	856
fflas_ftsv.inl	857
fflas_helpers.inl	857
igemm.doxy	858
igemm.h	858
igemm.inl	859
igemm_kernels.h	859
igemm_kernels.inl	860
igemm_tools.h	861
igemm_tools.inl	861
fflas_level1.inl	861
fflas_level2.inl	864
fflas_level3.inl	866
fflas_pfgemm.inl	869
fflas_pftsm.inl	870
fflas_simd.h	870
simd.doxy	872
simd128.inl	872
simd128_double.inl	873
simd128_float.inl	873
simd128_int16.inl	873
simd128_int32.inl	874
simd128_int64.inl	874
simd256.inl	875
simd256_double.inl	875
simd256_float.inl	876
simd256_int16.inl	876
simd256_int32.inl	877
simd256_int64.inl	877
simd512.inl	878
simd512_double.inl	878
simd512_float.inl	879
simd512_int32.inl	879
simd512_int64.inl	879
simd_modular.inl	880
fflas_sparse.h	880
fflas_sparse.inl	884
coo.h	886
coo_spm.inl	887
coo_spmv.inl	888
coo_utils.inl	889
csr.h	889
csr_pspm.inl	890

csr_pspmv.inl	891
csr_spmm.inl	891
csr_spmv.inl	892
csr_utils.inl	893
csr_hyb.h	894
csr_hyb_pspmm.inl	894
csr_hyb_pspmv.inl	895
csr_hyb_spmm.inl	895
csr_hyb_spmv.inl	896
csr_hyb_utils.inl	896
ell.h	897
ell_pspmm.inl	897
ell_pspmv.inl	898
ell_spmm.inl	899
ell_spmv.inl	900
ell_utils.inl	901
ell_simd.h	901
ell_simd_pspmv.inl	902
ell_simd_spmv.inl	902
ell_simd_utils.inl	903
hyb_zo.h	904
hyb_zo_pspmm.inl	904
hyb_zo_pspmv.inl	905
hyb_zo_spmm.inl	905
hyb_zo_spmv.inl	906
hyb_zo_utils.inl	906
read_sparse.h	907
sell.h	908
sell_pspmv.inl	908
sell_spmv.inl	909
sell_utils.inl	909
sparse_matrix_traits.h	910
utils.h	912
fflas_transpose.h	
Transpose the storage of the matrix (switch between row and col major mode)	912
ffpack.dox	913
ffpack.h	
Set of elimination based routines for dense linear algebra	913
ffpack.inl	922
ffpack_bruhatgen.inl	923
ffpack_charpoly.inl	925
ffpack_charpoly_danilevski.inl	925
ffpack_charpoly_kgfast.inl	926
ffpack_charpoly_kgfastgeneralized.inl	926
ffpack_charpoly_kglu.inl	927
ffpack_charpoly_mp.inl	927
ffpack_det_mp.inl	928
ffpack_echelonforms.inl	929
ffpack_fgesv.inl	930
ffpack_fgetrs.inl	930
ffpack_frobenius.inl	931
ffpack_fsytrf.inl	932
ffpack_ftrssyr2k.inl	933
ffpack_ftrstr.inl	933
ffpack_ftrtr.inl	934
ffpack_invert.inl	934
ffpack_krylovelim.inl	935
ffpack_ludivine.inl	935

ffpack_ludivine_mp.inl	936
ffpack_minpoly.inl	937
ffpack_permutation.inl	937
ffpack_pluq.inl	940
ffpack_pluq_mp.inl	940
ffpack_ppluq.inl	941
ffpack_rankprofiles.inl	942
field-traits.h	
Field Traits	943
field.doxy	945
rns-double-elt.h	
Rns elt structure with double support	945
rns-double-recint.inl	946
rns-double.h	
Rns structure with double support	946
rns-double.inl	947
rns-integer-mod.h	
Representation of $\mathbb{Z}/p\mathbb{Z}$ using RNS representation (note: fixed precision)	947
rns-integer.h	
Representation of \mathbb{Z} using RNS representation (note: fixed precision)	948
rns.h	949
rns.inl	949
interfaces.doxy	949
fflas_c.h	949
fflas_L1_inst.C	961
fflas_L1_inst.h	962
fflas_L1_inst_implem.inl	963
fflas_L2_inst.C	964
fflas_L2_inst.h	965
fflas_L2_inst_implem.inl	966
fflas_L3_inst.C	968
fflas_L3_inst.h	969
fflas_L3_inst_implem.inl	970
fflas_lv1.C	
C functions calls for level 1 FFLAS in flas-c.h	971
fflas_lv2.C	
C functions calls for level 2 FFLAS in flas-c.h	975
fflas_lv3.C	
C functions calls for level 3 FFLAS in flas-c.h	981
fflas_sparse.C	
C functions calls for level 1.5 and 2.5 FFLAS in flas-c.h	982
ffpack.C	
C functions calls for FFPACK in ffpack-c.h	983
ffpack_c.h	1003
ffpack_inst.C	1023
ffpack_inst.h	1024
ffpack_inst_implem.inl	1025
blockcuts.inl	1028
fflas_plevel1.h	1030
kaapi_routines.inl	1030
parallel.h	1030
pfgemm_variants.inl	1037
pfgemv.inl	1037
align-allocator.h	1038
args-parser.h	1038
bit_manipulation.h	1040
cast.h	1041

debug.h	
Various utilities for debugging	1041
fflas_intrinsic.h	1042
fflas_io.h	1042
fflas_memory.h	1043
fflas_randommatrix.h	1043
flimits.h	1045
Matio.h	1046
test-utils.h	1047
timer.h	1048
cblas.C	1048
clapack.C	1048
cuda.C	1049
fblas.C	1049
lapack.C	1050
regression-check.C	1051
test-charpoly-check.C	1052
test-charpoly.C	1052
test-compressQ.C	1053
test-det-check.C	1054
test-det.C	1055
test-echelon.C	1055
test-fadd.C	1058
test-fdot.C	1059
test-fgemm-check.C	1060
test-fgemm.C	1061
test-fgemv.C	1064
test-fger.C	1065
test-fgesv.C	1067
test-finit.C	1068
test-fscal.C	1069
test-fsyr2k.C	1071
test-fsyrrk.C	1072
test-fsytrf.C	1074
test-ftrmm.C	1075
test-ftrmv.C	1076
test-ftrsm-check.C	1078
test-ftrsm.C	1078
test-ftrssyr2k.C	1079
test-ftrstr.C	1081
test-ftrsv.C	1082
test-ftrtri.C	1083
test-interfaces-c.c	1084
test-invert-check.C	1084
test-io.C	1085
test-lu.C	1086
test-maxdelayeddim.C	1090
test-minpoly.C	1090
test-multifile1.C	1091
test-multifile2.C	1091
test-nullspace.C	1092
test-permutations.C	1093
test-pluq-check.C	1094
test-quasisep.C	1095
test-rankprofiles.C	1096
test-rpm.C	1097
test-simd.C	1097
test-solve.C	1101

test-storage-transpose.C	1101
101-fgemm.C	1102
2x2-fgemm.C	1102
2x2-ftrsv.C	1103
2x2-pluq.C	1103
fflas-101_1.C	1104
fflas-101_3.C	1104
fflas_101.C	1104
fflas_101_lvl1.C	1105
ffpack-fgesv.C	1105
ffpack-solve.C	1105

Chapter 10

Topic Documentation

10.1 CHECKER

Class CHECKER provides functions to verify computations in [FFLAS](#) and [FFPACK](#).

Class CHECKER provides functions to verify computations in [FFLAS](#) and [FFPACK](#).

10.2 FFLAS-FFPACK

the [FFLAS FFPACK](#) library

Modules

- [FFLAS](#)
The C-style wrapper of BLAS for finite field linear algebra.
- [Interfaces](#)
Interfaces for FFLAS-FFPACK.

10.2.1 Detailed Description

the [FFLAS FFPACK](#) library

C++ header library for fast exact dense linear algebra

See also

[FFLAS](#)
[FFPACK](#)

10.2.2 FFLAS

The C-style wrapper of BLAS for finite field linear algebra.

The C-style wrapper of BLAS for finite field linear algebra.

[FFLAS](#), Finite Field Linear Algebra Subroutines, provide basic linear algebra subroutines based on the BLAS interface. Therefore, the specifications are in C style; only the field given as a template parameter requires C++.

As much as possible, these routines use `ATLAS/BLAS` computations and achieve therefore high efficiency.

10.2.3 Interfaces

Interfaces for FFLAS-FFPACK.

Interfaces for FFLAS-FFPACK.

C interface in folder

See also

[libs](#)

10.3 Matrix Multiplication Algorithms

Matrix Multiplication (level 3) algorithms.

Files

- file [schedule_bini.inl](#)
Bini implementation.

10.3.1 Detailed Description

Matrix Multiplication (level 3) algorithms.

Todo biblio

10.4 SIMD wrapper

wraps SIMD functions Supportst SSE4.1, AVX, AVX2.

wraps SIMD functions Supportst SSE4.1, AVX, AVX2.

Todo biblio

10.5 FFPACK

Class [FFPACK](#) provides functions using fflas much as Lapack uses BLAS.

Namespaces

- namespace [FFPACK](#)
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

10.5.1 Detailed Description

Class [FFPACK](#) provides functions using fflas much as Lapack uses BLAS.

10.6 FFLAS-FFPACK fields

fields in the FFLAS-FFPACK library

Files

- file [rns-double-elt.h](#)
rns elt structure with double support
- file [rns-double.h](#)
rns structure with double support
- file [rns-integer-mod.h](#)
representation of $\mathbb{Z}/p\mathbb{Z}$ using RNS representation (note: fixed precision)
- file [rns-integer.h](#)
representation of \mathbb{Z} using RNS representation (note: fixed precision)
- file [rns.h](#)

10.6.1 Detailed Description

fields in the FFLAS-FFPACK library

Unparametric/Random elements

[Todo](#) biblio

10.7 RNS

just include them all

just include them all

Chapter 11

Namespace Documentation

11.1 FFLAS Namespace Reference

Namespaces

- namespace [_frtranspose_impl](#)
- namespace [BLAS3](#)
- namespace [csr_hyb_details](#)
- namespace [CuttingStrategy](#)
- namespace [details](#)
- namespace [details_spmv](#)
- namespace [ElementCategories](#)
- namespace [FieldCategories](#)

Traits and categories will need to be placed in a proper file later.

- namespace [MMHelperAlgo](#)
- namespace [ModeCategories](#)

Specifies the mode of action for an algorithm w.r.t.

- namespace [ParSeqHelper](#)

ParSeqHelper for both fgemm and ftrsm.

- namespace [Protected](#)
- namespace [sell_details](#)
- namespace [sparse_details](#)
- namespace [sparse_details_impl](#)
- namespace [StrategyParameter](#)
- namespace [StructureHelper](#)

StructureHelper for ftrsm.

- namespace [vectorised](#)

Data Structures

- struct [AlgoChooser](#)
- struct [AlgoChooser< ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeq >](#)
- struct [associatedDelayedField](#)
- struct [associatedDelayedField< const FFPACK::RNSIntegerMod< RNS > >](#)
- struct [associatedDelayedField< const Givaro::Modular< T, X > >](#)
- struct [associatedDelayedField< const Givaro::ModularBalanced< T > >](#)
- struct [associatedDelayedField< const Givaro::ZRing< T > >](#)

- struct [BlockTransposeSIMD](#)
- struct [Checker_Empty](#)
- class [CheckerImplem_fgemm](#)
- class [CheckerImplem_ftsm](#)
- struct [CooMat](#)
- struct [CsrMat](#)
- struct [ElementTraits](#)
 - *ElementTraits.*
- struct [ElementTraits< double >](#)
- struct [ElementTraits< FFPACK::rns_double_elt >](#)
- struct [ElementTraits< float >](#)
- struct [ElementTraits< Givaro::Integer >](#)
- struct [ElementTraits< int16_t >](#)
- struct [ElementTraits< int32_t >](#)
- struct [ElementTraits< int64_t >](#)
- struct [ElementTraits< int8_t >](#)
- struct [ElementTraits< Reclnt::rint< K > >](#)
- struct [ElementTraits< Reclnt::rmint< K, MG > >](#)
- struct [ElementTraits< Reclnt::ruint< K > >](#)
- struct [ElementTraits< uint16_t >](#)
- struct [ElementTraits< uint32_t >](#)
- struct [ElementTraits< uint64_t >](#)
- struct [ElementTraits< uint8_t >](#)
- struct [ElIMat](#)
- struct [FieldTraits](#)
 - *FieldTrait.*
- struct [FieldTraits< FFPACK::RNSInteger< T > >](#)
- struct [FieldTraits< FFPACK::RNSIntegerMod< T > >](#)
- struct [FieldTraits< Givaro::Modular< Element > >](#)
- struct [FieldTraits< Givaro::ModularBalanced< Element > >](#)
- struct [FieldTraits< Givaro::ZRing< double > >](#)
- struct [FieldTraits< Givaro::ZRing< float > >](#)
- struct [FieldTraits< Givaro::ZRing< Givaro::Integer > >](#)
- struct [FieldTraits< Givaro::ZRing< int16_t > >](#)
- struct [FieldTraits< Givaro::ZRing< int32_t > >](#)
- struct [FieldTraits< Givaro::ZRing< int64_t > >](#)
- struct [FieldTraits< Givaro::ZRing< Reclnt::ruint< K > > >](#)
- struct [FieldTraits< Givaro::ZRing< uint16_t > >](#)
- struct [FieldTraits< Givaro::ZRing< uint32_t > >](#)
- struct [FieldTraits< Givaro::ZRing< uint64_t > >](#)
- struct [has_minus_eq_impl](#)
- struct [has_minus_impl](#)
- struct [has_mul_eq_impl](#)
- struct [has_mul_impl](#)
- struct [has_operation](#)
- struct [has_plus_eq_impl](#)
- struct [has_plus_impl](#)
- struct [HelperFlag](#)
- struct [isSparseMatrix](#)
- struct [isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::COO > >](#)
- struct [isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::COO_ZO > >](#)
- struct [isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR > >](#)
- struct [isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR_HYB > >](#)
- struct [isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR_ZO > >](#)

- struct [isSparseMatrix](#)< Field, Sparse< Field, SparseMatrix_t::ELL > >
- struct [isSparseMatrix](#)< Field, Sparse< Field, SparseMatrix_t::ELL_simd > >
- struct [isSparseMatrix](#)< Field, Sparse< Field, SparseMatrix_t::ELL_simd_ZO > >
- struct [isSparseMatrix](#)< Field, Sparse< Field, SparseMatrix_t::ELL_ZO > >
- struct [isSparseMatrix](#)< Field, Sparse< Field, SparseMatrix_t::HYB_ZO > >
- struct [isSparseMatrix](#)< Field, Sparse< Field, SparseMatrix_t::SELL > >
- struct [isSparseMatrix](#)< Field, Sparse< Field, SparseMatrix_t::SELL_ZO > >
- struct [isSparseMatrixMKLFormat](#)
- struct [isSparseMatrixSimdFormat](#)
- struct [isZOSparseMatrix](#)
- struct [isZOSparseMatrix](#)< Field, Sparse< Field, SparseMatrix_t::COO_ZO > >
- struct [isZOSparseMatrix](#)< Field, Sparse< Field, SparseMatrix_t::CSR_ZO > >
- struct [isZOSparseMatrix](#)< Field, Sparse< Field, SparseMatrix_t::ELL_simd_ZO > >
- struct [isZOSparseMatrix](#)< Field, Sparse< Field, SparseMatrix_t::ELL_ZO > >
- struct [isZOSparseMatrix](#)< Field, Sparse< Field, SparseMatrix_t::SELL_ZO > >
- struct [MMHelper](#)
- struct [MMHelper](#)< FFPACK::RNSInteger< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >
- struct [MMHelper](#)< FFPACK::RNSIntegerMod< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >
- struct [MMHelper](#)< Field, AlgoTrait, ModeCategories::ConvertTo< Dest >, ParSeqTrait >
- struct [MMHelper](#)< Field, AlgoTrait, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeqTrait >
- struct [MMHelper](#)< Field, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >

FGEMM Helper for Default and ConvertTo modes of operation.

- struct [ModeTraits](#)
 - ModeTraits.*
 - struct [ModeTraits](#)< Givaro::Modular< Element, Compute > >
 - struct [ModeTraits](#)< Givaro::Modular< Givaro::Integer, Compute > >
 - struct [ModeTraits](#)< Givaro::Modular< int16_t, Compute > >
 - struct [ModeTraits](#)< Givaro::Modular< int32_t, Compute > >
 - struct [ModeTraits](#)< Givaro::Modular< int64_t, uint64_t > >
 - struct [ModeTraits](#)< Givaro::Modular< int8_t, Compute > >
 - struct [ModeTraits](#)< Givaro::Modular< Reclnt::ruint< K >, Compute > >
 - struct [ModeTraits](#)< Givaro::Modular< uint16_t, Compute > >
 - struct [ModeTraits](#)< Givaro::Modular< uint32_t, Compute > >
 - struct [ModeTraits](#)< Givaro::Modular< uint8_t, Compute > >
 - struct [ModeTraits](#)< Givaro::ModularBalanced< Element > >
 - struct [ModeTraits](#)< Givaro::ModularBalanced< Givaro::Integer > >
 - struct [ModeTraits](#)< Givaro::ModularBalanced< int16_t > >
 - struct [ModeTraits](#)< Givaro::ModularBalanced< int32_t > >
 - struct [ModeTraits](#)< Givaro::ModularBalanced< int8_t > >
 - struct [ModeTraits](#)< Givaro::Montgomery< T > >
 - struct [ModeTraits](#)< Givaro::ZRing< double > >
 - struct [ModeTraits](#)< Givaro::ZRing< float > >
 - struct [ModeTraits](#)< Givaro::ZRing< Givaro::Integer > >
 - struct [readMyMachineType](#)
 - struct [readMyMachineType](#)< Field, mpz_t >
 - struct [Sparse](#)
 - struct [Sparse](#)< _Field, SparseMatrix_t::COO >
 - struct [Sparse](#)< _Field, SparseMatrix_t::COO_ZO >
 - struct [Sparse](#)< _Field, SparseMatrix_t::CSR >
 - struct [Sparse](#)< _Field, SparseMatrix_t::CSR_HYB >
 - struct [Sparse](#)< _Field, SparseMatrix_t::CSR_ZO >
 - struct [Sparse](#)< _Field, SparseMatrix_t::ELL >
 - struct [Sparse](#)< _Field, SparseMatrix_t::ELL_simd >
 - struct [Sparse](#)< _Field, SparseMatrix_t::ELL_simd_ZO >

- struct [Sparse<_Field, SparseMatrix_t::ELL_ZO >](#)
- struct [Sparse<_Field, SparseMatrix_t::HYB_ZO >](#)
- struct [Sparse<_Field, SparseMatrix_t::SELL >](#)
- struct [Sparse<_Field, SparseMatrix_t::SELL_ZO >](#)
- struct [SpMat](#)
- struct [StatsMatrix](#)
- struct [support_fast_mod](#)
- struct [support_fast_mod< double >](#)
- struct [support_fast_mod< float >](#)
- struct [support_fast_mod< int64_t >](#)
- struct [support_simd](#)
- struct [support_simd_add](#)
- struct [support_simd_mod](#)
- struct [tfn_minus](#)
- struct [tfn_minus_eq](#)
- struct [tfn_mul](#)
- struct [tfn_mul_eq](#)
- struct [tfn_plus](#)
- struct [tfn_plus_eq](#)
- struct [TRSMHelper](#)

TRSM Helper.

Typedefs

- [template<class Field >](#)
[using Checker_fgemm = FFLAS::Checker_Empty< Field >](#)
- [template<class Field >](#)
[using Checker_ftrsm = FFLAS::Checker_Empty< Field >](#)
- [template<class Field >](#)
[using ForceCheck_fgemm = CheckerImplem_fgemm< Field >](#)
- [template<class Field >](#)
[using ForceCheck_ftrsm = CheckerImplem_ftrsm< Field >](#)
- [using ZOSparseMatrix = std::true_type](#)
- [using NotZOSparseMatrix = std::false_type](#)
- [using SimdSparseMatrix = std::true_type](#)
- [using NoSimdSparseMatrix = std::false_type](#)
- [using MKLSparseMatrixFormat = std::true_type](#)
- [using NotMKLSparseMatrixFormat = std::false_type](#)
- [template<class T >](#)
[using has_plus = typename std::conditional< std::is_arithmetic< T >::value, std::true_type, has_plus_impl< T > >::type](#)
- [template<class T >](#)
[using has_minus = typename std::conditional< std::is_arithmetic< T >::value, std::true_type, has_minus_impl< T > >::type](#)
- [template<class T >](#)
[using has_equal = typename std::conditional< std::is_arithmetic< T >::value, std::true_type, std::is_copyable< T > & _assignable< T > >::type](#)
- [template<class T >](#)
[using has_plus_eq = typename std::conditional< std::is_arithmetic< T >::value, std::true_type, has_plus_eq_impl< T > >::type](#)
- [template<class T >](#)
[using has_minus_eq = typename std::conditional< std::is_arithmetic< T >::value, std::true_type, has_minus_eq_impl< T > >::type](#)

- `template<class T >`
`using has_mul = typename std::conditional< std::is_arithmetic< T >::value, std::true_type, has_mul_impl<`
`T > >::type`
- `template<class T >`
`using has_mul_eq = typename std::conditional< std::is_arithmetic< T >::value, std::true_type,`
`has_mul_eq_impl< T > >::type`
- `typedef Givaro::Timer Timer`
- `typedef Givaro::BaseTimer BaseTimer`
- `typedef Givaro::UserTimer UserTimer`
- `typedef Givaro::SysTimer SysTimer`

Enumerations

- enum `FFLAS_ORDER` { `FflasRowMajor` = 101 , `FflasColMajor` = 102 }
Storage by row or col ?
- enum `FFLAS_TRANSPOSE` { `FflasNoTrans` = 111 , `FflasTrans` = 112 }
Is matrix transposed ?
- enum `FFLAS_UPLO` { `FflasUpper` = 121 , `FflasLower` = 122 , `FflasLeftTri` = 123 , `FflasRightTri` = 124 }
Is triangular matrix's shape upper ?
- enum `FFLAS_DIAG` { `FflasNonUnit` = 131 , `FflasUnit` = 132 }
Is the triangular matrix implicitly unit diagonal ?
- enum `FFLAS_SIDE` { `FflasLeft` = 141 , `FflasRight` = 142 }
On what side ?
- enum `FFLAS_BASE` { `FflasDouble` = 151 , `FflasFloat` = 152 , `FflasGeneric` = 153 }
FFLAS_BASE determines the type of the element representation for Matrix Mult kernel.
- enum `number_kind` { `zero` = 0 , `one` = 1 , `mone` = -1 , `other` = 2 }
- enum class `SparseMatrix_t` {
`CSR` , `CSR_ZO` , `CSC` , `CSC_ZO` ,
`COO` , `COO_ZO` , `ELL` , `ELL_ZO` ,
`SELL` , `SELL_ZO` , `ELL_simd` , `ELL_simd_ZO` ,
`CSR_HYB` , `HYB_ZO` }
- enum `FFLAS_FORMAT` {
`FflasAuto` = 0 , `FflasDense` = 1 , `FflasSMS` = 2 , `FflasBinary` = 3 ,
`FflasMath` = 4 , `FflasMaple` = 5 , `FflasSageMath` = 6 }

Functions

- `Givaro::Integer InfNorm (const size_t M, const size_t N, const Givaro::Integer *A, const size_t lda)`
- `template<class T >`
`const T & min3 (const T &m, const T &n, const T &k)`
- `template<class T >`
`const T & max3 (const T &m, const T &n, const T &k)`
- `template<class T >`
`const T & min4 (const T &m, const T &n, const T &k, const T &l)`
- `template<class T >`
`const T & max4 (const T &m, const T &n, const T &k, const T &l)`
- `template<class Field >`
`void fadd (const Field &F, const size_t N, typename Field::ConstElement_ptr A, const size_t inca, typename`
`Field::ConstElement_ptr B, const size_t incb, typename Field::Element_ptr C, const size_t incc)`
- `template<class Field >`
`void faddin (const Field &F, const size_t N, typename Field::ConstElement_ptr B, const size_t incb, typename`
`Field::Element_ptr C, const size_t incc)`

- `template<class Field >`
`void fsub (const Field &F, const size_t N, typename Field::ConstElement_ptr A, const size_t inca, typename Field::ConstElement_ptr B, const size_t incb, typename Field::Element_ptr C, const size_t incc)`
- `template<class Field >`
`void fsubin (const Field &F, const size_t N, typename Field::ConstElement_ptr B, const size_t incb, typename Field::Element_ptr C, const size_t incc)`
- `template<class Field >`
`void fadd (const Field &F, const size_t N, typename Field::ConstElement_ptr A, const size_t inca, const typename Field::Element alpha, typename Field::ConstElement_ptr B, const size_t incb, typename Field::Element_ptr C, const size_t incc)`
- `template<class Field >`
`void pfadd (const Field &F, const size_t M, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr C, const size_t ldc, const size_t numths)`
- `template<class Field >`
`void pfsub (const Field &F, const size_t M, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr C, const size_t ldc, const size_t numths)`
- `template<class Field >`
`void pfaddin (const Field &F, const size_t M, const size_t N, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr C, const size_t ldc, size_t numths)`
- `template<class Field >`
`void pfsubin (const Field &F, const size_t M, const size_t N, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr C, const size_t ldc, size_t numths)`
- `template<class Field >`
`void fadd (const Field &F, const size_t M, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr C, const size_t ldc)`
fadd : matrix addition.
- `template<class Field >`
`void fsub (const Field &F, const size_t M, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr C, const size_t ldc)`
fsub : matrix subtraction.
- `template<class Field >`
`void faddin (const Field &F, const size_t M, const size_t N, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr C, const size_t ldc)`
faddin
- `template<class Field >`
`void faddin (const Field &F, const FFLAS_UPLO uplo, const size_t N, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr C, const size_t ldc)`
fadding for symmetric matrices
- `template<class Field >`
`void fsubin (const Field &F, const size_t M, const size_t N, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr C, const size_t ldc)`
fsubin $C = C - B$
- `template<class Field >`
`void fadd (const Field &F, const size_t M, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, const typename Field::Element alpha, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr C, const size_t ldc)`
fadd : matrix addition with scaling.
- `template<class Field >`
`void fassign (const Field &F, const size_t N, typename Field::ConstElement_ptr Y, const size_t incY, typename Field::Element_ptr X, const size_t incX)`
fassign : $x \leftarrow y$.
- `template<> void fassign (const Givaro::Modular< float > &F, const size_t N, const float *Y, const size_t incY, float *X, const size_t incX)`

- `template<> void fassign (const Givaro::ModularBalanced< float > &F, const size_t N, const float *Y, const size_t incY, float *X, const size_t incX)`
- `template<> void fassign (const Givaro::ZRing< float > &F, const size_t N, const float *Y, const size_t incY, float *X, const size_t incX)`
- `template<> void fassign (const Givaro::Modular< double > &F, const size_t N, const double *Y, const size_t incY, double *X, const size_t incX)`
- `template<> void fassign (const Givaro::ModularBalanced< double > &F, const size_t N, const double *Y, const size_t incY, double *X, const size_t incX)`
- `template<> void fassign (const Givaro::ZRing< double > &F, const size_t N, const double *Y, const size_t incY, double *X, const size_t incX)`
- `template<class Field >`
`void fassign (const Field &F, const size_t m, const size_t n, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr A, const size_t lda)`

$$fassign : A \leftarrow B.$$
- `template<class Field >`
`void faxpy (const Field &F, const size_t N, const typename Field::Element alpha, typename Field::ConstElement_ptr X, const size_t incX, typename Field::Element_ptr Y, const size_t incY)`

$$faxpy : y \leftarrow \alpha \cdot x + y.$$
- `template<> void faxpy (const Givaro::DoubleDomain &, const size_t N, const Givaro::DoubleDomain::Element a, Givaro::DoubleDomain::ConstElement_ptr x, const size_t incx, Givaro::DoubleDomain::Element_ptr y, const size_t incy)`
- `template<> void faxpy (const Givaro::FloatDomain &, const size_t N, const Givaro::FloatDomain::Element a, Givaro::FloatDomain::ConstElement_ptr x, const size_t incx, Givaro::FloatDomain::Element_ptr y, const size_t incy)`
- `template<class Field >`
`void faxpy (const Field &F, const size_t m, const size_t n, const typename Field::Element alpha, typename Field::ConstElement_ptr X, const size_t ldx, typename Field::Element_ptr Y, const size_t ldy)`

$$faxpy : y \leftarrow \alpha \cdot x + y.$$
- `template<class Field >`
`Field::Element fdot (const Field &F, const size_t N, typename Field::ConstElement_ptr x, const size_t incx, typename Field::ConstElement_ptr y, const size_t incy, ModeCategories::DefaultTag &MT)`
- `template<class Field >`
`Field::Element fdot (const Field &F, const size_t N, typename Field::ConstElement_ptr x, const size_t incx, typename Field::ConstElement_ptr y, const size_t incy, ModeCategories::DelayedTag &MT)`
- `template<> Givaro::DoubleDomain::Element fdot (const Givaro::DoubleDomain &, const size_t N, Givaro::DoubleDomain::ConstElement_ptr x, const size_t incx, Givaro::DoubleDomain::ConstElement_ptr y, const size_t incy, ModeCategories::DefaultTag &MT)`
- `template<> Givaro::FloatDomain::Element fdot (const Givaro::FloatDomain &, const size_t N, Givaro::FloatDomain::ConstElement_ptr x, const size_t incx, Givaro::FloatDomain::ConstElement_ptr y, const size_t incy, ModeCategories::DefaultTag &MT)`
- `template<class Field, class T >`
`Field::Element fdot (const Field &F, const size_t N, typename Field::ConstElement_ptr x, const size_t incx, typename Field::ConstElement_ptr y, const size_t incy, ModeCategories::ConvertTo< T > &MT)`
- `template<class Field >`
`Field::Element fdot (const Field &F, const size_t N, typename Field::ConstElement_ptr x, const size_t incx, typename Field::ConstElement_ptr y, const size_t incy, ModeCategories::DefaultBoundedTag &dbt)`
- `template<class Field >`
`Field::Element fdot (const Field &F, const size_t N, typename Field::ConstElement_ptr x, const size_t incx, typename Field::ConstElement_ptr y, const size_t incy, const ParSeqHelper::Sequential seq)`
- `template<class Field >`
`Field::Element fdot (const Field &F, const size_t N, typename Field::ConstElement_ptr X, const size_t incX, typename Field::ConstElement_ptr Y, const size_t incY)`

$$fdot: dot\ product\ x^T y.$$
- `template<class Field >`
`Field::Element_ptr fgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, typename`

```
Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb,
const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper<
Field, MMHelperAlgo::Winograd, ModeCategories::ConvertTo< ElementCategories::MachineFloatTag >,
ParSeqHelper::Sequential > &H)
```

- `template<typename Field >`
`Field::Element_ptr fgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE`
`tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, typename`
`Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t`
`ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, const`
`ParSeqHelper::Sequential seq)`
- `template<typename Field , class Cut , class Param >`
`Field::Element_ptr fgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE`
`tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, typename`
`Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t`
`ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, const`
`ParSeqHelper::Parallel< Cut, Param > par)`
- `template<typename Field >`
`Field::Element_ptr fgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE`
`tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, typename`
`Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, const`
`typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc)`

*fgemm: Field **GE**neral **M**atrix **M**ultiply.*

- `template<typename Field , class ModeT , class ParSeq >`
`Field::Element_ptr fgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE`
`tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, typename`
`Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb,`
`const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field,`
`MMHelperAlgo::Auto, ModeT, ParSeq > &H)`
- `template<class Field >`
`Field::Element_ptr fgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE`
`tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, typename`
`Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb,`
`const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field,`
`MMHelperAlgo::Winograd, ModeCategories::DelayedTag, ParSeqHelper::Sequential > &H)`
- `template<class Field >`
`Field::Element_ptr fsquare (const Field &F, const FFLAS_TRANSPOSE ta, const size_t n, const typename`
`Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, const typename Field::Element`
`beta, typename Field::Element_ptr C, const size_t ldc)`

fsquare: Squares a matrix.

- `template<> double * fsquare (const Givaro::ModularBalanced< double > &F, const FFLAS_TRANSPOSE`
`ta, const size_t n, const double alpha, const double *A, const size_t lda, const double beta, double *C, const`
`size_t ldc)`
- `template<> float * fsquare (const Givaro::ModularBalanced< float > &F, const FFLAS_TRANSPOSE ta,`
`const size_t n, const float alpha, const float *A, const size_t lda, const float beta, float *C, const size_t ldc)`
- `template<> double * fsquare (const Givaro::Modular< double > &F, const FFLAS_TRANSPOSE ta, const`
`size_t n, const double alpha, const double *A, const size_t lda, const double beta, double *C, const size_t`
`ldc)`
- `template<> float * fsquare (const Givaro::Modular< float > &F, const FFLAS_TRANSPOSE ta, const size_t`
`n, const float alpha, const float *A, const size_t lda, const float beta, float *C, const size_t ldc)`
- `template<typename RNS , typename ParSeqTrait >`
`FFPACK::RNSInteger< RNS >::Element_ptr fgemm (const FFPACK::RNSInteger< RNS > &F, const`
`FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k,`
`const typename FFPACK::RNSInteger< RNS >::Element alpha, typename FFPACK::RNSInteger< RNS`
`>::ConstElement_ptr Ad, const size_t lda, typename FFPACK::RNSInteger< RNS >::ConstElement_ptr`
`Bd, const size_t ldb, const typename FFPACK::RNSInteger< RNS >::Element beta, typename`
`FFPACK::RNSInteger< RNS >::Element_ptr Cd, const size_t ldc, MMHelper< FFPACK::RNSInteger< RNS`
`>, MMHelperAlgo::Classic, ModeCategories::DefaultTag, ParSeqHelper::Compose< ParSeqHelper::Sequential,`
`ParSeqTrait > > &H)`

- `template<typename RNS >`
`FFPACK::RNSInteger< RNS >::Element_ptr fgemm (const FFPACK::RNSInteger< RNS > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename FFPACK::RNSInteger< RNS >::Element alpha, typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Ad, const size_t lda, typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Bd, const size_t ldb, const typename FFPACK::RNSInteger< RNS >::Element beta, typename FFPACK::RNSInteger< RNS >::Element_ptr Cd, const size_t ldc, MMHelper< FFPACK::RNSInteger< RNS >, MMHelperAlgo::Classic, ModeCategories::DefaultTag, ParSeqHelper::Sequential > &H)`
- `template<typename RNS, typename ParSeqTrait >`
`FFPACK::RNSInteger< RNS >::Element_ptr fgemm (const FFPACK::RNSInteger< RNS > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename FFPACK::RNSInteger< RNS >::Element alpha, typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Ad, const size_t lda, typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Bd, const size_t ldb, const typename FFPACK::RNSInteger< RNS >::Element beta, typename FFPACK::RNSInteger< RNS >::Element_ptr Cd, const size_t ldc, MMHelper< FFPACK::RNSInteger< RNS >, MMHelperAlgo::Classic, ModeCategories::DefaultTag, ParSeqHelper::Compose< ParSeqHelper::Parallel< CuttingStrategy::RNSModulus, StrategyParameter::Threads >, ParSeqTrait > > &H)`
- `template<typename RNS, typename Cut, typename Param >`
`FFPACK::RNSInteger< RNS >::Element_ptr fgemm (const FFPACK::RNSInteger< RNS > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename FFPACK::RNSInteger< RNS >::Element alpha, typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Ad, const size_t lda, typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Bd, const size_t ldb, const typename FFPACK::RNSInteger< RNS >::Element beta, typename FFPACK::RNSInteger< RNS >::Element_ptr Cd, const size_t ldc, MMHelper< FFPACK::RNSInteger< RNS >, MMHelperAlgo::Classic, ModeCategories::DefaultTag, ParSeqHelper::Parallel< Cut, Param > > &H)`
- `template<class ParSeq >`
`Givaro::Integer * fgemm (const Givaro::ZRing< Givaro::Integer > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const Givaro::Integer alpha, const Givaro::Integer *A, const size_t lda, const Givaro::Integer *B, const size_t ldb, Givaro::Integer beta, Givaro::Integer *C, const size_t ldc, MMHelper< Givaro::ZRing< Givaro::Integer >, MMHelperAlgo::Classic, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeq > &H)`
- `template<typename RNS, class ModeT >`
`RNS::Element_ptr fgemm (const FFPACK::RNSInteger< RNS > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename RNS::Element alpha, typename RNS::ConstElement_ptr Ad, const size_t lda, typename RNS::ConstElement_ptr Bd, const size_t ldb, const typename RNS::Element beta, typename RNS::Element_ptr Cd, const size_t ldc, MMHelper< FFPACK::RNSInteger< RNS >, MMHelperAlgo::Winograd, ModeT, ParSeqHelper::Sequential > &H)`
- `template<typename RNS >`
`RNS::Element_ptr fgemm (const FFPACK::RNSIntegerMod< RNS > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename RNS::Element alpha, typename RNS::ConstElement_ptr Ad, const size_t lda, typename RNS::ConstElement_ptr Bd, const size_t ldb, const typename RNS::Element beta, typename RNS::Element_ptr Cd, const size_t ldc, MMHelper< FFPACK::RNSIntegerMod< RNS >, MMHelperAlgo::Winograd > &H)`
- `Givaro::Integer * fgemm (const Givaro::Modular< Givaro::Integer > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const Givaro::Integer alpha, const Givaro::Integer *A, const size_t lda, const Givaro::Integer *B, const size_t ldb, const Givaro::Integer beta, Givaro::Integer *C, const size_t ldc, MMHelper< Givaro::Modular< Givaro::Integer >, MMHelperAlgo::Classic, ModeCategories::ConvertTo< ElementCategories::RNSElementTag > > &H)`
- `template<class ParSeq >`
`Givaro::Integer * fgemm (const Givaro::Modular< Givaro::Integer > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const Givaro::Integer alpha, const Givaro::Integer *A, const size_t lda, const Givaro::Integer *B, const size_t ldb, const Givaro::Integer beta, Givaro::Integer *C, const size_t ldc, MMHelper< Givaro::Modular< Givaro::Integer >, MMHelperAlgo::Auto, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeq > &H)`
- `template<size_t K1, size_t K2, class ParSeq >`
`Reclnt::ruint< K1 > * fgemm (const Givaro::Modular< Reclnt::ruint< K1 >, Reclnt::ruint< K2 > > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n,`

- ```
const size_t k, const RecInt::ruint< K1 > alpha, const RecInt::ruint< K1 > *A, const size_t lda, const
RecInt::ruint< K1 > *B, const size_t ldb, RecInt::ruint< K1 > beta, RecInt::ruint< K1 > *C, const size_t
ldc, MMHelper< Givaro::Modular< RecInt::ruint< K1 >, RecInt::ruint< K2 > >, MMHelperAlgo::Classic,
ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeq > &H)
```
- `template<class Field , class ModeT >`  
`Field::Element_ptr fgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE`  
`tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, typename`  
`Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb,`  
`const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field,`  
`MMHelperAlgo::Winograd, ModeT > &H)`
  - `template<class Field , class ModeT , class Cut , class Param >`  
`Field::Element_ptr fgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE`  
`tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, typename`  
`Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb,`  
`const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field,`  
`MMHelperAlgo::WinogradPar, ModeT, ParSeqHelper::Parallel< Cut, Param > > &H)`
  - `template<class Field >`  
`Field::Element_ptr fgemv (const Field &F, const FFLAS_TRANSPOSE ta, const size_t M, const size_t`  
`N, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda,`  
`typename Field::ConstElement_ptr X, const size_t incX, const typename Field::Element beta, typename`  
`Field::Element_ptr Y, const size_t incY, MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::ConvertTo<`  
`ElementCategories::MachineFloatTag > > &H)`
  - `template<class Field >`  
`Field::Element_ptr fgemv (const Field &F, const FFLAS_TRANSPOSE ta, const size_t M, const size_t`  
`N, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda,`  
`typename Field::ConstElement_ptr X, const size_t incX, const typename Field::Element beta, typename`  
`Field::Element_ptr Y, const size_t incY, MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::DelayedTag`  
`> &H)`
  - `template<class Field >`  
`Field::Element_ptr fgemv (const Field &F, const FFLAS_TRANSPOSE ta, const size_t M, const size_t`  
`N, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda,`  
`typename Field::ConstElement_ptr X, const size_t incX, const typename Field::Element beta, typename`  
`Field::Element_ptr Y, const size_t incY, MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::DefaultTag`  
`> &H)`
  - `template<class Field >`  
`Field::Element_ptr fgemv (const Field &F, const FFLAS_TRANSPOSE ta, const size_t M, const size_t`  
`N, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda,`  
`typename Field::ConstElement_ptr X, const size_t incX, const typename Field::Element beta, typename`  
`Field::Element_ptr Y, const size_t incY, MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::LazyTag`  
`> &H)`
  - `template<class Field >`  
`Field::Element_ptr fgemv (const Field &F, const FFLAS_TRANSPOSE TransA, const size_t M, const`  
`size_t N, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda,`  
`typename Field::ConstElement_ptr X, const size_t incX, const typename Field::Element beta, typename`  
`Field::Element_ptr Y, const size_t incY)`
- finite prime Field GEneral Matrix Vector multiplication.*
- `Givaro::ZRing< int64_t >::Element_ptr fgemv (const Givaro::ZRing< int64_t > &F, const FFLAS_TRANSPOSE`  
`ta, const size_t M, const size_t N, const int64_t alpha, const int64_t *A, const size_t lda, const int64_t *X,`  
`const size_t incX, const int64_t beta, int64_t *Y, const size_t incY, MMHelper< Givaro::ZRing< int64_t > ,`  
`MMHelperAlgo::Classic, ModeCategories::DefaultTag > &H)`
  - `Givaro::DoubleDomain::Element_ptr fgemv (const Givaro::DoubleDomain &F, const FFLAS_TRANSPOSE`  
`ta, const size_t M, const size_t N, const Givaro::DoubleDomain::Element alpha, const Givaro::Double`  
`Domain::ConstElement_ptr A, const size_t lda, const Givaro::DoubleDomain::ConstElement_ptr X, const`  
`size_t incX, const Givaro::DoubleDomain::Element beta, Givaro::DoubleDomain::Element_ptr Y, const size_t`  
`incY, MMHelper< Givaro::DoubleDomain, MMHelperAlgo::Classic, ModeCategories::DefaultTag > &H)`
  - `template<class Field >`  
`Field::Element_ptr fgemv (const Field &F, const FFLAS_TRANSPOSE ta, const size_t M, const size_t`

- N, const typename Field::Element alpha, const typename Field::ConstElement\_ptr A, const size\_t lda, const typename Field::ConstElement\_ptr X, const size\_t incX, const typename Field::Element beta, typename Field::Element\_ptr Y, const size\_t incY, MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::DefaultBoundedTag > &H)
- Givaro::FloatDomain::Element\_ptr fgemv (const Givaro::FloatDomain &F, const FFLAS\_TRANSPOSE ta, const size\_t M, const size\_t N, const Givaro::FloatDomain::Element alpha, const Givaro::FloatDomain::ConstElement\_ptr A, const size\_t lda, const Givaro::FloatDomain::ConstElement\_ptr X, const size\_t incX, const Givaro::FloatDomain::Element beta, Givaro::FloatDomain::Element\_ptr Y, const size\_t incY, MMHelper< Givaro::FloatDomain, MMHelperAlgo::Classic, ModeCategories::DefaultTag > &H)
  - template<class Field, class Cut, class Param >  
Field::Element\_ptr fgemv (const Field &F, const FFLAS\_TRANSPOSE ta, const size\_t m, const size\_t n, const typename Field::Element alpha, const typename Field::ConstElement\_ptr A, const size\_t lda, const typename Field::ConstElement\_ptr X, const size\_t incX, const typename Field::Element beta, typename Field::Element\_ptr Y, const size\_t incY, ParSeqHelper::Parallel< Cut, Param > &parH)
  - template<class Field >  
Field::Element\_ptr fgemv (const Field &F, const FFLAS\_TRANSPOSE ta, const size\_t m, const size\_t n, const typename Field::Element alpha, const typename Field::ConstElement\_ptr A, const size\_t lda, const typename Field::ConstElement\_ptr X, const size\_t incX, const typename Field::Element beta, typename Field::Element\_ptr Y, const size\_t incY, ParSeqHelper::Sequential &seqH)
  - FFPACK::rns\_double::Element\_ptr fgemv (const FFPACK::RNSInteger< FFPACK::rns\_double > &F, const FFLAS\_TRANSPOSE ta, const size\_t M, const size\_t N, const FFPACK::rns\_double::Element alpha, FFPACK::rns\_double::ConstElement\_ptr A, const size\_t lda, FFPACK::rns\_double::ConstElement\_ptr X, const size\_t incX, const FFPACK::rns\_double::Element beta, FFPACK::rns\_double::Element\_ptr Y, const size\_t incY, MMHelper< FFPACK::RNSInteger< FFPACK::rns\_double >, MMHelperAlgo::Classic, ModeCategories::DefaultTag > &H)
  - FFPACK::rns\_double::Element\_ptr fgemv (const FFPACK::RNSIntegerMod< FFPACK::rns\_double > &F, const FFLAS\_TRANSPOSE ta, const size\_t M, const size\_t N, const FFPACK::rns\_double::Element alpha, FFPACK::rns\_double::ConstElement\_ptr A, const size\_t lda, FFPACK::rns\_double::ConstElement\_ptr X, const size\_t incX, const FFPACK::rns\_double::Element beta, FFPACK::rns\_double::Element\_ptr Y, const size\_t incY, MMHelper< FFPACK::RNSIntegerMod< FFPACK::rns\_double >, MMHelperAlgo::Classic, ModeCategories::DefaultTag > &H)
  - Givaro::Integer \* fgemv (const Givaro::ZRing< Givaro::Integer > &F, const FFLAS\_TRANSPOSE ta, const size\_t m, const size\_t n, const Givaro::Integer alpha, Givaro::Integer \*A, const size\_t lda, Givaro::Integer \*X, const size\_t ldx, Givaro::Integer beta, Givaro::Integer \*Y, const size\_t ldy, MMHelper< Givaro::ZRing< Givaro::Integer >, MMHelperAlgo::Classic, ModeCategories::ConvertTo< ElementCategories::RNSElementTag > > &H)
  - Givaro::Integer \* fgemv (const Givaro::Modular< Givaro::Integer > &F, const FFLAS\_TRANSPOSE ta, const size\_t m, const size\_t n, const Givaro::Integer alpha, Givaro::Integer \*A, const size\_t lda, Givaro::Integer \*X, const size\_t ldx, Givaro::Integer beta, Givaro::Integer \*Y, const size\_t ldy, MMHelper< Givaro::Modular< Givaro::Integer >, MMHelperAlgo::Classic, ModeCategories::ConvertTo< ElementCategories::RNSElementTag > > &H)
  - template<size\_t K1, size\_t K2, class ParSeq >  
Reclnt::ruint< K1 > \* fgemv (const Givaro::Modular< Reclnt::ruint< K1 >, Reclnt::ruint< K2 > > &F, const FFLAS\_TRANSPOSE ta, const size\_t m, const size\_t n, const Reclnt::ruint< K1 > alpha, const Reclnt::ruint< K1 > \*A, const size\_t lda, const Reclnt::ruint< K1 > \*X, const size\_t incx, Reclnt::ruint< K1 > beta, Reclnt::ruint< K1 > \*Y, const size\_t incy, MMHelper< Givaro::Modular< Reclnt::ruint< K1 >, Reclnt::ruint< K2 > >, MMHelperAlgo::Classic, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeq > &H)
  - template<class Field >  
void fger (const Field &F, const size\_t M, const size\_t N, const typename Field::Element alpha, typename Field::ConstElement\_ptr x, const size\_t incx, typename Field::ConstElement\_ptr y, const size\_t incy, typename Field::Element\_ptr A, const size\_t lda)  
*fger: rank one update of a general matrix*
  - template<class Field >  
void fger (const Field &F, const size\_t M, const size\_t N, const typename Field::Element alpha, typename Field::ConstElement\_ptr x, const size\_t incx, typename Field::ConstElement\_ptr y, const size\_t incy, typename Field::Element\_ptr A, const size\_t lda, MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::ConvertTo< ElementCategories::MachineFloatTag > > &H)



- `template<class Field , class AnyTag >`  
`void fger (const Field &F, const size_t M, const size_t N, const typename Field::Element alpha, typename Field::ConstElement_ptr x, const size_t incx, typename Field::ConstElement_ptr y, const size_t incy, typename Field::Element_ptr A, const size_t lda, MMHelper< Field, MMHelperAlgo::Classic, AnyTag > &H)`
- `void fger (const Givaro::DoubleDomain &F, const size_t M, const size_t N, const Givaro::DoubleDomain::↔ Element alpha, const Givaro::DoubleDomain::ConstElement_ptr x, const size_t incx, const Givaro::Double↔ Domain::ConstElement_ptr y, const size_t incy, Givaro::DoubleDomain::Element_ptr A, const size_t lda, MMHelper< Givaro::DoubleDomain, MMHelperAlgo::Classic, ModeCategories::DefaultTag > &H)`
- `template<class Field >`  
`void fger (const Field &F, const size_t M, const size_t N, const typename Field::Element alpha, const typename Field::ConstElement_ptr x, const size_t incx, const typename Field::ConstElement_ptr y, const size_t incy, typename Field::Element_ptr A, const size_t lda, MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::DefaultBoundedTag > &H)`
- `void fger (const Givaro::FloatDomain &F, const size_t M, const size_t N, const Givaro::FloatDomain::Element alpha, const Givaro::FloatDomain::ConstElement_ptr x, const size_t incx, const Givaro::FloatDomain::↔ ConstElement_ptr y, const size_t incy, Givaro::FloatDomain::Element_ptr A, const size_t lda, MMHelper< Givaro::FloatDomain, MMHelperAlgo::Classic, ModeCategories::DefaultTag > &H)`
- `template<class Field >`  
`void fger (const Field &F, const size_t M, const size_t N, const typename Field::Element alpha, typename Field::ConstElement_ptr x, const size_t incx, typename Field::ConstElement_ptr y, const size_t incy, typename Field::Element_ptr A, const size_t lda, MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::LazyTag > &H)`
- `template<class Field >`  
`void fger (const Field &F, const size_t M, const size_t N, const typename Field::Element alpha, typename Field::ConstElement_ptr x, const size_t incx, typename Field::ConstElement_ptr y, const size_t incy, typename Field::Element_ptr A, const size_t lda, MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::DelayedTag > &H)`
- `void fger (const Givaro::Modular< Givaro::Integer > &F, const size_t M, const size_t N, const typename Givaro::Integer alpha, typename Givaro::Integer *x, const size_t incx, typename Givaro::Integer *y, const size_t incy, typename Givaro::Integer *A, const size_t lda, MMHelper< Givaro::Modular< Givaro::Integer >, MMHelperAlgo::Classic, ModeCategories::ConvertTo< ElementCategories::RNSElementTag > > &H)`
- `template<typename RNS >`  
`void fger (const FFPACK::RNSInteger< RNS > &F, const size_t M, const size_t N, const typename FFPACK::RNSInteger< RNS >::Element alpha, typename FFPACK::RNSInteger< RNS >::Element_ptr x, const size_t incx, typename FFPACK::RNSInteger< RNS >::Element_ptr y, const size_t incy, typename FFPACK::RNSInteger< RNS >::Element_ptr A, const size_t lda, MMHelper< FFPACK::RNSInteger< RNS >, MMHelperAlgo::Classic, ModeCategories::DefaultTag > &H)`
- `template<typename RNS >`  
`void fger (const FFPACK::RNSIntegerMod< RNS > &F, const size_t M, const size_t N, const typename FFPACK::RNSIntegerMod< RNS >::Element alpha, typename FFPACK::RNSIntegerMod< RNS >::Element_ptr x, const size_t incx, typename FFPACK::RNSIntegerMod< RNS >::Element_ptr y, const size_t incy, typename FFPACK::RNSIntegerMod< RNS >::Element_ptr A, const size_t lda, MMHelper< FFPACK::RNSIntegerMod< RNS >, MMHelperAlgo::Classic > &H)`
- `template<class Field >`  
`void freduce (const Field &F, const size_t n, typename Field::ConstElement_ptr Y, const size_t incY, typename Field::Element_ptr X, const size_t incX)`  

$$freduce\ x \leftarrow ymodF.$$
- `template<class Field >`  
`void freduce (const Field &F, const size_t n, typename Field::Element_ptr X, const size_t incX)`  

$$freduce\ x \leftarrow xmodF.$$
- `template<class Field >`  
`void freduce_constoverride (const Field &F, const size_t m, typename Field::ConstElement_ptr A, const size_t incX)`
- `template<class Field , class ConstOtherElement_ptr >`  
`void finit (const Field &F, const size_t n, ConstOtherElement_ptr Y, const size_t incY, typename Field::Element_ptr X, const size_t incX)`

- `template<class Field >`  
`void finit (const Field &F, const size_t n, typename Field::Element_ptr X, const size_t incX)`  
*finit Initializes  $X$  in  $F^{\$}$ .*
- `template<class Field >`  
`void freduce (const Field &F, const size_t m, const size_t n, typename Field::Element_ptr A, const size_t lda)`  
*freduce  $A \leftarrow A \bmod F$ .*
- `template<class Field >`  
`void freduce (const Field &F, const FFLAS_UPLO uplo, const size_t N, typename Field::Element_ptr A, const size_t lda)`  
*freduce for square symmetric matrices*
- `template<class Field >`  
`void pfreduce (const Field &F, const size_t m, const size_t n, typename Field::Element_ptr A, const size_t lda, const size_t numths)`
- `template<class Field >`  
`void freduce (const Field &F, const size_t m, const size_t n, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr A, const size_t lda)`  
*freduce  $A \leftarrow B \bmod F$ .*
- `template<class Field >`  
`void freduce_constoverride (const Field &F, const size_t m, const size_t n, typename Field::ConstElement_ptr A, const size_t lda)`
- `template<class Field, class OtherElement_ptr >`  
`void finit (const Field &F, const size_t m, const size_t n, const OtherElement_ptr B, const size_t ldb, typename Field::Element_ptr A, const size_t lda)`  
*finit  $A \leftarrow B \bmod F$ .*
- `template<class Field >`  
`void finit (const Field &F, const size_t m, const size_t n, typename Field::Element_ptr A, const size_t lda)`
- `template<> void freduce (const FFPACK::RNSIntegerMod< FFPACK::rns_double > &F, const size_t n, FFPACK::RNSIntegerMod< FFPACK::rns_double >::Element_ptr A, size_t inc)`
- `template<> void freduce (const FFPACK::RNSIntegerMod< FFPACK::rns_double > &F, const size_t m, const size_t n, FFPACK::rns_double::Element_ptr A, size_t lda)`
- `template<class Field >`  
`bool freivalds (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::ConstElement_ptr C, const size_t ldc)`  
*freivalds: **FE**reivalds **GE**neral **M**atrix **M**ultiply **R**andom **C**heck.*
- `template<class Field >`  
`void fscal (const Field &F, const size_t n, const typename Field::Element alpha, typename Field::Element_ptr X, const size_t incX)`  
*fscal  $x \leftarrow \alpha \cdot x$ .*
- `template<class Field >`  
`void fscal (const Field &F, const size_t n, const typename Field::Element alpha, typename Field::ConstElement_ptr X, const size_t incX, typename Field::Element_ptr Y, const size_t incY)`  
*fscal  $y \leftarrow \alpha \cdot x$ .*
- `template<> void fscal (const Givaro::DoubleDomain &, const size_t N, const Givaro::DoubleDomain::Element a, Givaro::DoubleDomain::ConstElement_ptr x, const size_t incx, Givaro::DoubleDomain::Element_ptr y, const size_t incy)`
- `template<> void fscal (const Givaro::FloatDomain &, const size_t N, const Givaro::FloatDomain::Element a, Givaro::FloatDomain::ConstElement_ptr x, const size_t incx, Givaro::FloatDomain::Element_ptr y, const size_t incy)`
- `template<> void fscal (const Givaro::DoubleDomain &, const size_t N, const Givaro::DoubleDomain::Element a, Givaro::DoubleDomain::Element_ptr y, const size_t incy)`
- `template<> void fscal (const Givaro::FloatDomain &, const size_t N, const Givaro::FloatDomain::Element a, Givaro::FloatDomain::Element_ptr y, const size_t incy)`

- `template<class Field >`  
`void fscaln (const Field &F, const size_t m, const size_t n, const typename Field::Element alpha, typename Field::Element_ptr A, const size_t lda)`  

$$fscaln\ A \leftarrow a \cdot A.$$
- `template<class Field >`  
`void fscal (const Field &F, const size_t m, const size_t n, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb)`  

$$fscal\ B \leftarrow a \cdot A.$$
- `template<> void fscaln (const FFPACK::RNSInteger< FFPACK::rns_double > &F, const size_t n, const FFPACK::rns_double::Element alpha, FFPACK::rns_double::Element_ptr A, const size_t inc)`
- `template<> void fscal (const FFPACK::RNSInteger< FFPACK::rns_double > &F, const size_t n, const FFPACK::rns_double::Element alpha, FFPACK::rns_double::ConstElement_ptr A, const size_t Ainc, FFPACK::rns_double::Element_ptr B, const size_t Binc)`
- `template<> void fscaln (const FFPACK::RNSInteger< FFPACK::rns_double > &F, const size_t m, const size_t n, const FFPACK::rns_double::Element alpha, FFPACK::rns_double::Element_ptr A, const size_t lda)`
- `template<> void fscal (const FFPACK::RNSInteger< FFPACK::rns_double > &F, const size_t m, const size_t n, const FFPACK::rns_double::Element alpha, FFPACK::rns_double::ConstElement_ptr A, const size_t lda, FFPACK::rns_double::Element_ptr B, const size_t ldb)`
- `template<> void fscaln (const FFPACK::RNSIntegerMod< FFPACK::rns_double > &F, const size_t n, const typename FFPACK::RNSIntegerMod< FFPACK::rns_double >::Element alpha, typename FFPACK::RNSIntegerMod< FFPACK::rns_double >::Element_ptr A, const size_t inc)`
- `template<> void fscal (const FFPACK::RNSIntegerMod< FFPACK::rns_double > &F, const size_t n, const FFPACK::rns_double::Element alpha, FFPACK::rns_double::ConstElement_ptr A, const size_t Ainc, FFPACK::rns_double::Element_ptr B, const size_t Binc)`
- `template<> void fscaln (const FFPACK::RNSIntegerMod< FFPACK::rns_double > &F, const size_t m, const size_t n, const FFPACK::rns_double::Element alpha, FFPACK::rns_double::Element_ptr A, const size_t lda)`
- `template<> void fscal (const FFPACK::RNSIntegerMod< FFPACK::rns_double > &F, const size_t m, const size_t n, const FFPACK::rns_double::Element alpha, FFPACK::rns_double::ConstElement_ptr A, const size_t lda, FFPACK::rns_double::Element_ptr B, const size_t ldb)`
- `template<class Field >`  
`Field::Element_ptr fsyr2k (const Field &F, const FFLAS_UPLO UpLo, const FFLAS_TRANSPOSE trans, const size_t n, const size_t k, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc)`  

$$fsyr2k: \text{Symmetric Rank } 2K \text{ update}$$
- `template<class Field >`  
`Field::Element_ptr fsyrk (const Field &F, const FFLAS_UPLO UpLo, const FFLAS_TRANSPOSE trans, const size_t n, const size_t k, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc)`  

$$fsyrk: \text{Symmetric Rank } K \text{ update}$$
- `template<class Field >`  
`Field::Element_ptr fsyrk (const Field &F, const FFLAS_UPLO UpLo, const FFLAS_TRANSPOSE trans, const size_t N, const size_t K, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, const ParSeqHelper::Sequential seq)`
- `template<class Field >`  
`Field::Element_ptr fsyrk (const Field &F, const FFLAS_UPLO UpLo, const FFLAS_TRANSPOSE trans, const size_t N, const size_t K, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::DefaultTag > &H)`
- `template<class Field >`  
`Field::Element_ptr fsyrk (const Field &F, const FFLAS_UPLO UpLo, const FFLAS_TRANSPOSE trans, const size_t N, const size_t K, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::ConvertTo< ElementCategories::MachineFloatTag >, ParSeqHelper::Sequential > &H)`



- `template<class Field >`  
`Field::Element_ptr fsyrk (const Field &F, const FFLAS_UPLO UpLo, const FFLAS_TRANSPOSE trans, const size_t N, const size_t K, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::DelayedTag > &H)`
- `template<class Field >`  
`Field::Element_ptr fsyrk (const Field &F, const FFLAS_UPLO UpLo, const FFLAS_TRANSPOSE trans, const size_t N, const size_t K, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::LazyTag > &H)`
- `template<class Field , typename Mode >`  
`Field::Element_ptr fsyrk (const Field &F, const FFLAS_UPLO UpLo, const FFLAS_TRANSPOSE trans, const size_t N, const size_t K, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field, MMHelperAlgo::DivideAndConquer, Mode > &H)`
- `template<class Field >`  
`Field::Element_ptr fsyrk (const Field &F, const FFLAS_UPLO UpLo, const FFLAS_TRANSPOSE trans, const size_t N, const size_t K, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::DefaultBoundedTag > &H)`
- `Givaro::FloatDomain::Element_ptr fsyrk (const Givaro::FloatDomain &F, const FFLAS_UPLO UpLo, const FFLAS_TRANSPOSE trans, const size_t N, const size_t K, const Givaro::FloatDomain::Element alpha, Givaro::FloatDomain::ConstElement_ptr A, const size_t lda, const Givaro::FloatDomain::Element beta, Givaro::FloatDomain::Element_ptr C, const size_t ldc, MMHelper< Givaro::FloatDomain, MMHelperAlgo::Classic, ModeCategories::DefaultTag > &H)`
- `Givaro::DoubleDomain::Element_ptr fsyrk (const Givaro::DoubleDomain &F, const FFLAS_UPLO UpLo, const FFLAS_TRANSPOSE trans, const size_t N, const size_t K, const Givaro::DoubleDomain::Element alpha, Givaro::DoubleDomain::ConstElement_ptr A, const size_t lda, const Givaro::DoubleDomain::Element beta, Givaro::DoubleDomain::Element_ptr C, const size_t ldc, MMHelper< Givaro::DoubleDomain, MMHelperAlgo::Classic, ModeCategories::DefaultTag > &H)`
- `template<class Field >`  
`Field::Element_ptr fsyrk (const Field &F, const FFLAS_UPLO UpLo, const FFLAS_TRANSPOSE trans, const size_t n, const size_t k, const typename Field::Element alpha, typename Field::Element_ptr A, const size_t lda, typename Field::ConstElement_ptr D, const size_t incD, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, const size_t threshold=__FFLASFFPACK_FSYRK_THRESHOLD)`  
*fsyrk: Symmetric Rank K update with diagonal scaling*
- `template<class Field >`  
`Field::Element_ptr fsyrk (const Field &F, const FFLAS_UPLO UpLo, const FFLAS_TRANSPOSE trans, const size_t N, const size_t K, const typename Field::Element alpha, typename Field::Element_ptr A, const size_t lda, typename Field::ConstElement_ptr D, const size_t incD, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, const ParSeqHelper::Sequential seq, const size_t threshold)`
- `template<class Field , class Cut , class Param >`  
`Field::Element_ptr fsyrk (const Field &F, const FFLAS_UPLO UpLo, const FFLAS_TRANSPOSE trans, const size_t N, const size_t K, const typename Field::Element alpha, typename Field::Element_ptr A, const size_t lda, typename Field::ConstElement_ptr D, const size_t incD, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, const ParSeqHelper::Parallel< Cut, Param > par, const size_t threshold)`
- `template<class Field >`  
`Field::Element_ptr fsyrk (const Field &F, const FFLAS_UPLO UpLo, const FFLAS_TRANSPOSE trans, const size_t n, const size_t k, const typename Field::Element alpha, typename Field::Element_ptr A, const size_t lda, typename Field::ConstElement_ptr D, const size_t incD, const std::vector< bool > &twoBlock, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, const size_t threshold=__FFLASFFPACK_FSYRK_THRESHOLD)`  
*fsyrk: Symmetric Rank K update with diagonal scaling*
- `template<class Field , class FieldTrait >`  
`void computeS1S2 (const Field &F, const FFLAS_TRANSPOSE trans, const size_t N, const size_t K, const typename Field::Element x, const typename Field::Element y, typename Field::Element_ptr A, const`

- size\_t Ida, typename Field::Element\_ptr S, const size\_t Ids, typename Field::Element\_ptr T, const size\_t Idt, MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > &WH)
- template<class Field >  
Field::Element\_ptr fsyrk (const Field &F, const FFLAS\_UPLO UpLo, const FFLAS\_TRANSPOSE trans, const size\_t N, const size\_t K, const typename Field::Element alpha, typename Field::ConstElement\_ptr A, const size\_t Ida, const typename Field::Element beta, typename Field::Element\_ptr C, const size\_t Idc, MMHelper< Field, MMHelperAlgo::Winograd, ModeCategories::DelayedTag, ParSeqHelper::Sequential > &H)
  - template<class Field , class Mode >  
Field::Element\_ptr fsyrk (const Field &F, const FFLAS\_UPLO UpLo, const FFLAS\_TRANSPOSE trans, const size\_t N, const size\_t K, const typename Field::Element alpha, typename Field::ConstElement\_ptr A, const size\_t Ida, const typename Field::Element beta, typename Field::Element\_ptr C, const size\_t Idc, MMHelper< Field, MMHelperAlgo::Winograd, Mode > &H)
  - template<class Field , class FieldTrait >  
Field::Element\_ptr fsyrk\_strassen (const Field &F, const FFLAS\_UPLO uplo, const FFLAS\_TRANSPOSE trans, const size\_t N, const size\_t K, const typename Field::Element y1, const typename Field::Element y2, const typename Field::Element alpha, typename Field::Element\_ptr A, const size\_t Ida, const typename Field::Element beta, typename Field::Element\_ptr C, const size\_t Idc, MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > &WH)
  - template<class Field >  
void ftrmm (const Field &F, const FFLAS\_SIDE Side, const FFLAS\_UPLO Uplo, const FFLAS\_TRANSPOSE TransA, const FFLAS\_DIAG Diag, const size\_t M, const size\_t N, const typename Field::Element alpha, typename Field::ConstElement\_ptr A, const size\_t Ida, typename Field::Element\_ptr B, const size\_t ldb)  
*ftrmm: **TR**angular **M**atrix **M**ultiply.*
  - template<class Field >  
void ftrmm (const Field &F, const FFLAS\_SIDE Side, const FFLAS\_UPLO Uplo, const FFLAS\_TRANSPOSE TransA, const FFLAS\_DIAG Diag, const size\_t M, const size\_t N, const typename Field::Element alpha, typename Field::ConstElement\_ptr A, const size\_t Ida, typename Field::ConstElement\_ptr B, const size\_t ldb, const typename Field::Element beta, typename Field::Element\_ptr C, const size\_t ldc)  
*ftrmm: **TR**angular **M**atrix **M**ultiply with 3 operands Computes  $C \leftarrow \alpha \text{op}(A)B + \text{beta}C$  or  $C \leftarrow \alpha \text{Bop}(A) + \text{beta}C$ .*
  - template<class Field >  
void ftrsm (const Field &F, const FFLAS\_SIDE Side, const FFLAS\_UPLO Uplo, const FFLAS\_TRANSPOSE TransA, const FFLAS\_DIAG Diag, const size\_t M, const size\_t N, const typename Field::Element alpha, typename Field::Element\_ptr A, const size\_t Ida, typename Field::Element\_ptr B, const size\_t ldb)
  - template<class Field >  
void ftrsm (const Field &F, const FFLAS\_SIDE Side, const FFLAS\_UPLO Uplo, const FFLAS\_TRANSPOSE TransA, const FFLAS\_DIAG Diag, const size\_t M, const size\_t N, const typename Field::Element alpha, typename Field::Element\_ptr A, const size\_t Ida, typename Field::Element\_ptr B, const size\_t ldb, const ParSeqHelper::Sequential &PSH)
  - template<class Field , class Cut , class Param >  
void ftrsm (const Field &F, const FFLAS\_SIDE Side, const FFLAS\_UPLO Uplo, const FFLAS\_TRANSPOSE TransA, const FFLAS\_DIAG Diag, const size\_t M, const size\_t N, const typename Field::Element alpha, typename Field::Element\_ptr A, const size\_t Ida, typename Field::Element\_ptr B, const size\_t ldb, const ParSeqHelper::Parallel< Cut, Param > &PSH)
  - template<class Field , class ParSeqTrait = ParSeqHelper::Sequential>  
void ftrsm (const Field &F, const FFLAS\_SIDE Side, const FFLAS\_UPLO Uplo, const FFLAS\_TRANSPOSE TransA, const FFLAS\_DIAG Diag, const size\_t M, const size\_t N, const typename Field::Element alpha, typename Field::Element\_ptr A, const size\_t Ida, typename Field::Element\_ptr B, const size\_t ldb, TRSMHelper< StructureHelper::Recursive, ParSeqTrait > &H)
  - void ftrsm (const Givaro::Modular< Givaro::Integer > &F, const FFLAS\_SIDE Side, const FFLAS\_UPLO Uplo, const FFLAS\_TRANSPOSE TransA, const FFLAS\_DIAG Diag, const size\_t M, const size\_t N, const Givaro::Integer alpha, const Givaro::Integer \*A, const size\_t Ida, Givaro::Integer \*B, const size\_t ldb)
  - void cblas\_imptrsm (const enum FFLAS\_ORDER Order, const enum FFLAS\_SIDE Side, const enum FFLAS\_UPLO Uplo, const enum FFLAS\_TRANSPOSE TransA, const enum FFLAS\_DIAG Diag, const int M, const int N, const FFPACK::rns\_double\_elt alpha, FFPACK::rns\_double\_elt\_cstptr A, const int Ida, FFPACK::rns\_double\_elt\_ptr B, const int ldb)
  - template<class Field >  
void ftrsv (const Field &F, const FFLAS\_UPLO Uplo, const FFLAS\_TRANSPOSE TransA, const FFLAS\_DIAG

Diag, const size\_t N, typename Field::ConstElement\_ptr A, const size\_t lda, typename Field::Element\_ptr X, int incX)

*trsv: TRIangular System solve with Vector Computes  $X \leftarrow \text{op}(A^{-1})X$*

- void igemm\_ (const enum FFLAS\_ORDER Order, const enum FFLAS\_TRANSPOSE TransA, const enum FFLAS\_TRANSPOSE TransB, const size\_t M, const size\_t N, const size\_t K, const int64\_t alpha, const int64\_t \*A, const size\_t lda, const int64\_t \*B, const size\_t ldb, const int64\_t beta, int64\_t \*C, const size\_t ldc)
- template<class Field , class OtherElement\_ptr >  
void finit (const Field &F, const size\_t n, const OtherElement\_ptr Y, const size\_t incY, typename Field::Element\_ptr X, const size\_t incX)  
*finit  $x \leftarrow y \bmod F$ .*
- template<class Field , class OtherElement\_ptr >  
void fconvert (const Field &F, const size\_t n, OtherElement\_ptr X, const size\_t incX, typename Field::ConstElement\_ptr Y, const size\_t incY)  
*fconvert  $x \leftarrow y \bmod F$ .*
- template<class Field >  
void fnegin (const Field &F, const size\_t n, typename Field::Element\_ptr X, const size\_t incX)  
*fnegin  $x \leftarrow -x$ .*
- template<class Field >  
void fneg (const Field &F, const size\_t n, typename Field::ConstElement\_ptr Y, const size\_t incY, typename Field::Element\_ptr X, const size\_t incX)  
*fneg  $x \leftarrow -y$ .*
- template<class Field >  
void fzero (const Field &F, const size\_t n, typename Field::Element\_ptr X, const size\_t incX)  
*fzero :  $A \leftarrow 0$ .*
- template<class Field , class RandIter >  
void frand (const Field &F, RandIter &G, const size\_t n, typename Field::Element\_ptr X, const size\_t incX)  
*frand :  $A \leftarrow \text{random}$ .*
- template<class Field >  
bool fiszero (const Field &F, const size\_t n, typename Field::ConstElement\_ptr X, const size\_t incX)  
*fiszero : test  $X = 0$ .*
- template<class Field >  
bool fequal (const Field &F, const size\_t n, typename Field::ConstElement\_ptr X, const size\_t incX, typename Field::ConstElement\_ptr Y, const size\_t incY)  
*fequal : test  $X = Y$ .*
- template<class Field >  
void faxpby (const Field &F, const size\_t N, const typename Field::Element alpha, typename Field::ConstElement\_ptr X, const size\_t incX, const typename Field::Element beta, typename Field::Element\_ptr Y, const size\_t incY)  
*faxpby :  $y \leftarrow \alpha \cdot x + \beta \cdot y$ .*
- template<typename Field , class Cut , class Param >  
Field::Element fdot (const Field &F, const size\_t N, typename Field::ConstElement\_ptr X, const size\_t incX, typename Field::ConstElement\_ptr Y, const size\_t incY, const ParSeqHelper::Parallel< Cut, Param > par)
- template<class Field >  
void fswap (const Field &F, const size\_t N, typename Field::Element\_ptr X, const size\_t incX, typename Field::Element\_ptr Y, const size\_t incY)  
*fswap :  $X \leftrightarrow Y$ .*
- template<class Field >  
void fzero (const Field &F, const size\_t m, const size\_t n, typename Field::Element\_ptr A, const size\_t lda)  
*fzero :  $A \leftarrow 0$ .*
- template<class Field >  
void fzero (const Field &F, const FFLAS\_UPLO shape, const FFLAS\_DIAG diag, const size\_t n, typename Field::Element\_ptr A, const size\_t lda)  
*fzero :  $A \leftarrow 0$  for a triangular matrix.*

- `template<class Field , class RandIter >`  
`void frand (const Field &F, RandIter &G, const size_t m, const size_t n, typename Field::Element_ptr A, const size_t lda)`  

$$frand : A \leftarrow random.$$
- `template<class Field >`  
`bool fequal (const Field &F, const size_t m, const size_t n, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb)`  

$$fequal : test A = B.$$
- `template<class Field >`  
`bool fiszero (const Field &F, const size_t m, const size_t n, typename Field::ConstElement_ptr A, const size_t lda)`  

$$fiszero : test A = 0.$$
- `template<class Field >`  
`void fidentity (const Field &F, const size_t m, const size_t n, typename Field::Element_ptr A, const size_t lda, const typename Field::Element &d)`  

$$creates a diagonal matrix$$
- `template<class Field >`  
`void fidentity (const Field &F, const size_t m, const size_t n, typename Field::Element_ptr A, const size_t lda)`  

$$creates a diagonal matrix$$
- `template<class Field , class OtherElement_ptr >`  
`void finit (const Field &F, const size_t m, const size_t n, typename Field::Element_ptr A, const size_t lda)`  

$$finit \text{ Initializes } A \text{ in } F^{\$}.$$
- `template<class Field , class OtherElement_ptr >`  
`void fconvert (const Field &F, const size_t m, const size_t n, OtherElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb)`  

$$fconvert A \leftarrow B \bmod F.$$
- `template<class Field >`  
`void fnegin (const Field &F, const size_t m, const size_t n, typename Field::Element_ptr A, const size_t lda)`  

$$fnegin A \leftarrow -A.$$
- `template<class Field >`  
`void fneg (const Field &F, const size_t m, const size_t n, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::Element_ptr A, const size_t lda)`  

$$fneg A \leftarrow -B.$$
- `template<class Field >`  
`void faxpby (const Field &F, const size_t m, const size_t n, const typename Field::Element alpha, typename Field::ConstElement_ptr X, const size_t ldx, const typename Field::Element beta, typename Field::Element_ptr Y, const size_t ldy)`  

$$faxpby : y \leftarrow \alpha \cdot x + \beta \cdot y.$$
- `template<class Field >`  
`void fmove (const Field &F, const size_t m, const size_t n, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb)`  

$$fmove : A \leftarrow B \text{ and } B \leftarrow 0.$$
- `template<class Field >`  
`size_t bitsize (const Field &F, size_t M, size_t N, const typename Field::ConstElement_ptr A, size_t lda)`  

$$bitsize : \text{Computes the largest bitsize of the matrix' coefficients.}$$
- `template<> size_t bitsize< Givaro::ZRing< Givaro::Integer > > (const Givaro::ZRing< Givaro::Integer >`  
`&F, size_t M, size_t N, const Givaro::Integer *A, size_t lda)`
- `template<class Field >`  
`void ftrmv (const Field &F, const FFLAS_UPLO Uplo, const FFLAS_TRANSPOSE TransA, const FFLAS_DIAG Diag, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr X, int incX)`  

$$ftrsm : TRIangular Matrix Vector prodcut \text{ Computes } X \leftarrow op(A)X$$

- `template<class Field >`  
`void ftrsm (const Field &F, const FFLAS_SIDE Side, const FFLAS_UPLO Uplo, const FFLAS_TRANSPOSE TransA, const FFLAS_DIAG Diag, const size_t M, const size_t N, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb)`  
*ftrsm: **TR**iangular **S**ystem solve with **M**atrix.*
- `template<class Field , typename FieldTrait >`  
`Field::Element_ptr fsyrk_strassen (const Field &F, const FFLAS_UPLO UpLo, const FFLAS_TRANSPOSE trans, const size_t N, const size_t K, const typename Field::Element y1, const typename Field::Element y2, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > &H)`
- `template<typename Field >`  
`Field::Element_ptr pfgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, size_t numthreads=0)`
- `template<class Field >`  
`Field::Element * pfgemm_1D_rec (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, const typename Field::Element_ptr A, const size_t lda, const typename Field::Element_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element *C, const size_t ldc, size_t seuil)`
- `template<class Field >`  
`Field::Element * pfgemm_2D_rec (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, const typename Field::Element_ptr A, const size_t lda, const typename Field::Element_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element *C, const size_t ldc, size_t seuil)`
- `template<class Field >`  
`Field::Element * pfgemm_3D_rec (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, const typename Field::Element_ptr A, const size_t lda, const typename Field::Element_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, size_t seuil, size_t *x)`
- `template<class Field >`  
`Field::Element_ptr pfgemm_3D_rec2 (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, const typename Field::Element_ptr A, const size_t lda, const typename Field::Element_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, size_t seuil, size_t *x)`
- `template<class Field , class ModeTrait , class Strat , class Param >`  
`std::enable_if<!std::is_same< ModeTrait, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >::value, typename Field::Element_ptr >::type fgemm (const Field &F, const FFLAS::FFLAS_TRANSPOSE ta, const FFLAS::FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field, MMHelperAlgo::Winograd, ModeTrait, ParSeqHelper::Parallel< Strat, Param > > &H)`
- `template<class Field , class Cut , class Param >`  
`Field::Element_ptr ftrsm (const Field &F, const FFLAS::FFLAS_SIDE Side, const FFLAS::FFLAS_UPLO UpLo, const FFLAS::FFLAS_TRANSPOSE TA, const FFLAS::FFLAS_DIAG Diag, const size_t m, const size_t n, const typename Field::Element alpha, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb, TRSMHelper< StructureHelper::Iterative, ParSeqHelper::Parallel< Cut, Param > > &H)`
- `template<class Field , class Cut , class Param >`  
`Field::Element_ptr ftrsm (const Field &F, const FFLAS::FFLAS_SIDE Side, const FFLAS::FFLAS_UPLO UpLo, const FFLAS::FFLAS_TRANSPOSE TA, const FFLAS::FFLAS_DIAG Diag, const size_t m, const size_t n, const typename Field::Element alpha, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb, TRSMHelper< StructureHelper::Hybrid, ParSeqHelper::Parallel< Cut, Param > > &H)`

- `template<class Field , class SM >`  
`void fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, const typename Field::Element &beta, typename Field::Element_ptr y)`
- `template<class Field , class SM >`  
`void fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int Idx, const typename Field::Element &beta, typename Field::Element_ptr y, int Idy)`
- `template<class Field , class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::COO > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class Field , class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::COO_ZO > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::COO > &A)`
- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::COO_ZO > &A)`
- `template<class Field , class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::CSR > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class Field , class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::CSR_ZO > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::CSR > &A)`
- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::CSR_ZO > &A)`
- `template<class Field >`  
`std::ostream & sparse_print (std::ostream &os, const Sparse< Field, SparseMatrix_t::CSR > &A)`
- `template<class IndexT >`  
`void sparse_init (const Givaro::Modular< Givaro::Integer > &F, Sparse< Givaro::Modular< Givaro::Integer >, SparseMatrix_t::CSR > &A, const IndexT *row, const IndexT *col, Givaro::Integer *dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class IndexT >`  
`void sparse_init (const Givaro::ZRing< Givaro::Integer > &F, Sparse< Givaro::ZRing< Givaro::Integer >, SparseMatrix_t::CSR_ZO > &A, const IndexT *row, const IndexT *col, Givaro::Integer *dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class IndexT , size_t RECINT_SIZE>`  
`void sparse_init (const Givaro::ZRing< RecInt::rmint< RECINT_SIZE > > &F, Sparse< Givaro::ZRing< RecInt::rmint< RECINT_SIZE > >, SparseMatrix_t::CSR_ZO > &A, const IndexT *row, const IndexT *col, typename Givaro::ZRing< RecInt::rmint< RECINT_SIZE > >::Element_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class IndexT , size_t RECINT_SIZE>`  
`void sparse_init (const Givaro::ZRing< RecInt::rmint< RECINT_SIZE > > &F, Sparse< Givaro::ZRing< RecInt::rmint< RECINT_SIZE > >, SparseMatrix_t::CSR > &A, const IndexT *row, const IndexT *col, typename Givaro::ZRing< RecInt::rmint< RECINT_SIZE > >::Element_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::CSR_HYB > &A)`
- `template<class Field , class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::CSR_HYB > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class Field , class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::ELL > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class Field , class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::ELL_ZO > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`



- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::ELL > &A)`
- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::ELL_ZO > &A)`
- `template<class Field , class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::ELL_simd > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class Field , class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::ELL_simd > &A)`
- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A)`
- `template<class Field >`  
`void sparse_print (const Sparse< Field, SparseMatrix_t::ELL_simd > &A)`
- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::HYB_ZO > &A)`
- `template<class Field , class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::HYB_ZO > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<typename _Field >`  
`std::ostream & operator<< (std::ostream &os, const Sparse< _Field, SparseMatrix_t::HYB_ZO > &A)`
- `template<class Field , bool sorted = true, bool read_integer = false>`  
`void readSmsFormat (const std::string &path, const Field &f, index_t *&row, index_t *&col, typename Field::Element_ptr &val, index_t &rowdim, index_t &coldim, uint64_t &nnz)`
- `template<class Field >`  
`void readSprFormat (const std::string &path, const Field &f, index_t *&row, index_t *&col, typename Field::Element_ptr &val, index_t &rowdim, index_t &coldim, uint64_t &nnz)`
- `template<class T >`  
`std::enable_if< std::is_integral< T >::value, int > getDataType ()`
- `template<class T >`  
`std::enable_if< std::is_floating_point< T >::value, int > getDataType ()`
- `template<class T >`  
`std::enable_if< std::is_same< T, mpz_t >::value, int > getDataType ()`
- `template<class T >`  
`int getDataType ()`
- `template<class Field >`  
`void readMachineType (const Field &F, typename Field::Element &modulo, typename Field::Element_ptr val, std::ifstream &file, const uint64_t dims, const mask_t data_type, const mask_t field_desc)`
- `template<class Field >`  
`void readDnsFormat (const std::string &path, const Field &F, index_t &rowdim, index_t &coldim, typename Field::Element_ptr &val)`
- `template<class Field >`  
`void writeDnsFormat (const std::string &path, const Field &F, const index_t &rowdim, const index_t &coldim, typename Field::Element_ptr A, index_t ldA)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::SELL_ZO > &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::ModularTag)`
- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::SELL > &A)`
- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::SELL_ZO > &A)`
- `template<class Field >`  
`void sparse_print (const Sparse< Field, SparseMatrix_t::SELL > &A)`

- `template<class Field , class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::SELL > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz, uint64_t sigma=0)`
- `template<class Field , class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::SELL_ZO > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class It >`  
`double computeDeviation (It begin, It end)`
- `template<class Field >`  
`StatsMatrix getStat (const Field &F, const index_t *row, const index_t *col, typename Field::ConstElement_ptr val, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class Field , class enable = void>`  
`Field::Residu_t maxCardinality ()`
- `template<> uint64_t maxCardinality< Givaro::Modular< int64_t > > ()`
- `template<> uint32_t maxCardinality< Givaro::Modular< int32_t > > ()`
- `template<class Field >`  
`Field::Residu_t minCardinality ()`
- `template<> void fflas_delete (FFPACK::rns_double_elt_ptr A)`
- `template<> void fflas_delete (FFPACK::rns_double_elt_cstptr A)`
- `template<> FFPACK::rns_double_elt_ptr fflas_new (const FFPACK::RNSIntegerMod< FFPACK::rns_double > &F, const size_t m, const Alignment align)`
- `template<> FFPACK::rns_double_elt_ptr fflas_new (const FFPACK::RNSIntegerMod< FFPACK::rns_double > &F, const size_t m, const size_t n, const Alignment align)`
- `template<typename RNS >`  
`void finit_rns (const FFPACK::RNSIntegerMod< RNS > &F, const size_t m, const size_t n, size_t k, const Givaro::Integer *B, const size_t ldb, typename RNS::Element_ptr A)`
- `template<typename RNS >`  
`void finit_trans_rns (const FFPACK::RNSIntegerMod< RNS > &F, const size_t m, const size_t n, size_t k, const Givaro::Integer *B, const size_t ldb, typename RNS::Element_ptr A)`
- `template<typename RNS >`  
`void fconvert_rns (const FFPACK::RNSIntegerMod< RNS > &F, const size_t m, const size_t n, Givaro::Integer alpha, Givaro::Integer *B, const size_t ldb, typename RNS::ConstElement_ptr A)`
- `template<typename RNS >`  
`void fconvert_trans_rns (const FFPACK::RNSIntegerMod< RNS > &F, const size_t m, const size_t n, Givaro::Integer alpha, Givaro::Integer *B, const size_t ldb, typename RNS::ConstElement_ptr A)`
- `template<> FFPACK::rns_double_elt_ptr fflas_new (const FFPACK::RNSInteger< FFPACK::rns_double > &F, const size_t m, const Alignment align)`
- `template<> FFPACK::rns_double_elt_ptr fflas_new (const FFPACK::RNSInteger< FFPACK::rns_double > &F, const size_t m, const size_t n, const Alignment align)`
- `template<typename RNS >`  
`void finit_rns (const FFPACK::RNSInteger< RNS > &F, const size_t m, const size_t n, size_t k, const Givaro::Integer *B, const size_t ldb, typename FFPACK::RNSInteger< RNS >::Element_ptr A)`
- `template<typename RNS >`  
`void fconvert_rns (const FFPACK::RNSInteger< RNS > &F, const size_t m, const size_t n, Givaro::Integer alpha, Givaro::Integer *B, const size_t ldb, typename FFPACK::RNSInteger< RNS >::ConstElement_ptr A)`
- `template INST_OR_DECL void freduce (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t n, const FFLAS_ELT *X, const size_t incX)`  

$$\text{freduce } x \leftarrow x \bmod F.$$
- `template INST_OR_DECL void freduce (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t n, const FFLAS_ELT *Y, const size_t incY, FFLAS_ELT *X, const size_t incX)`  

$$\text{freduce } x \leftarrow y \bmod F.$$
- `template INST_OR_DECL void finit (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t n, const FFLAS_ELT *Y, const size_t incY, FFLAS_ELT *X, const size_t incX)`  

$$\text{finit } x \leftarrow y \bmod F.$$



- `template INST_OR_DECL void fconvert (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t n, FFLAS_ELT *X, const size_t incX, const FFLAS_ELT *Y, const size_t incY)`  
 $fconvert\ x \leftarrow y \bmod F.$
- `template INST_OR_DECL void fnegin (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t n, FFLAS_ELT *X, const size_t incX)`  
 $fnegin\ x \leftarrow -x.$
- `template INST_OR_DECL void fneg (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t n, const FFLAS_ELT *Y, const size_t incY, FFLAS_ELT *X, const size_t incX)`  
 $fneg\ x \leftarrow -y.$
- `template INST_OR_DECL void fzero (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t n, FFLAS_ELT *X, const size_t incX)`  
 $fzero : A \leftarrow 0.$
- `template INST_OR_DECL bool fiszero (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t n, const FFLAS_ELT *X, const size_t incX)`  
 $fiszero : test\ X = 0.$
- `template INST_OR_DECL bool fequal (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t n, const FFLAS_ELT *X, const size_t incX, const FFLAS_ELT *Y, const size_t incY)`  
 $fequal : test\ X = Y.$
- `template INST_OR_DECL void fassign (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t N, const FFLAS_ELT *Y, const size_t incY, FFLAS_ELT *X, const size_t incX)`  
 $fassign : x \leftarrow y.$
- `template INST_OR_DECL void fscaln (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t n, const FFLAS_ELT alpha, FFLAS_ELT *X, const size_t incX)`  
 $fscaln\ x \leftarrow \alpha \cdot x.$
- `template INST_OR_DECL void fscal (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t n, const FFLAS_ELT alpha, const FFLAS_ELT *X, const size_t incX, FFLAS_ELT *Y, const size_t incY)`  
 $fscal\ y \leftarrow \alpha \cdot x.$
- `template INST_OR_DECL void faxpy (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t N, const FFLAS_ELT alpha, const FFLAS_ELT *X, const size_t incX, FFLAS_ELT *Y, const size_t incY)`  
 $faxpy : y \leftarrow \alpha \cdot x + y.$
- `template INST_OR_DECL FFLAS_ELT fdot (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t N, const FFLAS_ELT *X, const size_t incX, const FFLAS_ELT *Y, const size_t incY)`  
 $faxpby : y \leftarrow \alpha \cdot x + \beta \cdot y.$
- `template INST_OR_DECL void fswap (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t N, FFLAS_ELT *X, const size_t incX, FFLAS_ELT *Y, const size_t incY)`  
 $fswap : X \leftrightarrow Y.$
- `template INST_OR_DECL void fadd (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t N, const FFLAS_ELT *A, const size_t inca, const FFLAS_ELT *B, const size_t incb, FFLAS_ELT *C, const size_t incc)`
- `template INST_OR_DECL void fsub (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t N, const FFLAS_ELT *A, const size_t inca, const FFLAS_ELT *B, const size_t incb, FFLAS_ELT *C, const size_t incc)`
- `template INST_OR_DECL void faddn (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t N, const FFLAS_ELT *B, const size_t incb, FFLAS_ELT *C, const size_t incc)`
- `template INST_OR_DECL void fadd (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t N, const FFLAS_ELT *A, const size_t inca, const FFLAS_ELT alpha, const FFLAS_ELT *B, const size_t incb, FFLAS_ELT *C, const size_t incc)`
- `template INST_OR_DECL void fassign (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t m, const size_t n, const FFLAS_ELT *B, const size_t ldb, FFLAS_ELT *A, const size_t lda)`  
 $fassign : A \leftarrow B.$
- `template INST_OR_DECL void fzero (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t m, const size_t n, FFLAS_ELT *A, const size_t lda)`  
 $fzero : A \leftarrow 0.$

- `template INST_OR_DECL bool fequal (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t m, const size_t n, const FFLAS_ELT *A, const size_t lda, const FFLAS_ELT *B, const size_t ldb)`  
 $fequal : test A = B.$
- `template INST_OR_DECL bool fiszero (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t m, const size_t n, const FFLAS_ELT *A, const size_t lda)`  
 $fiszero : test A = 0.$
- `template INST_OR_DECL void fidentity (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t m, const size_t n, FFLAS_ELT *A, const size_t lda, const FFLAS_ELT &d)`  
 $creates a diagonal matrix$
- `template INST_OR_DECL void fidentity (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t m, const size_t n, FFLAS_ELT *A, const size_t lda)`  
 $creates a diagonal matrix$
- `template INST_OR_DECL void freduce (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t m, const size_t n, FFLAS_ELT *A, const size_t lda)`  
 $freduce A \leftarrow A mod F.$
- `template INST_OR_DECL void freduce (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t m, const size_t n, const FFLAS_ELT *B, const size_t ldb, FFLAS_ELT *A, const size_t lda)`  
 $freduce A \leftarrow B mod F.$
- `template INST_OR_DECL void finit (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t m, const size_t n, const FFLAS_ELT *B, const size_t ldb, FFLAS_ELT *A, const size_t lda)`  
 $finit A \leftarrow B mod F.$
- `template INST_OR_DECL void fnegin (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t m, const size_t n, FFLAS_ELT *A, const size_t lda)`  
 $fnegin A \leftarrow -A.$
- `template INST_OR_DECL void fneg (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t m, const size_t n, const FFLAS_ELT *B, const size_t ldb, FFLAS_ELT *A, const size_t lda)`  
 $fneg A \leftarrow -B.$
- `template INST_OR_DECL void fscaln (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t m, const size_t n, const FFLAS_ELT alpha, FFLAS_ELT *A, const size_t lda)`  
 $fscaln A \leftarrow a \cdot A.$
- `template INST_OR_DECL void fscal (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t m, const size_t n, const FFLAS_ELT alpha, const FFLAS_ELT *A, const size_t lda, FFLAS_ELT *B, const size_t ldb)`  
 $fscal B \leftarrow a \cdot A.$
- `template INST_OR_DECL void faxpy (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t m, const size_t n, const FFLAS_ELT alpha, const FFLAS_ELT *X, const size_t ldx, FFLAS_ELT *Y, const size_t ldy)`  
 $faxpy : y \leftarrow \alpha \cdot x + y.$
- `template INST_OR_DECL void fmove (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t m, const size_t n, FFLAS_ELT *A, const size_t lda, FFLAS_ELT *B, const size_t ldb)`  
 $faxpby : y \leftarrow \alpha \cdot x + \beta \cdot y.$
- `template INST_OR_DECL void fadd (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t M, const size_t N, const FFLAS_ELT *A, const size_t lda, const FFLAS_ELT *B, const size_t ldb, FFLAS_ELT *C, const size_t ldc)`  
 $fadd : matrix addition.$
- `template INST_OR_DECL void fsub (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t M, const size_t N, const FFLAS_ELT *A, const size_t lda, const FFLAS_ELT *B, const size_t ldb, FFLAS_ELT *C, const size_t ldc)`  
 $fsub : matrix subtraction.$
- `template INST_OR_DECL void fsubin (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t M, const size_t N, const FFLAS_ELT *B, const size_t ldb, FFLAS_ELT *C, const size_t ldc)`  
 $fsubin C = C - B$
- `template INST_OR_DECL void fadd (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t M, const size_t N, const FFLAS_ELT *A, const size_t lda, const FFLAS_ELT alpha, const FFLAS_ELT *B, const size_t ldb, FFLAS_ELT *C, const size_t ldc)`

*fadd* : matrix addition with scaling.

- `template INST_OR_DECL void faddin (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t M, const size_t N, const FFLAS_ELT *B, const size_t ldb, FFLAS_ELT *C, const size_t ldc)`

*faddin*

- `template INST_OR_DECL FFLAS_ELT * fgemv (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS_TRANSPOSE TransA, const size_t M, const size_t N, const FFLAS_ELT alpha, const FFLAS_ELT *A, const size_t lda, const FFLAS_ELT *X, const size_t incX, const FFLAS_ELT beta, FFLAS_ELT *Y, const size_t incY)`

*finite prime FFLAS\_FIELD<FFLAS\_ELT> GEneral Matrix Vector multiplication.*

- `template INST_OR_DECL void fger (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t M, const size_t N, const FFLAS_ELT alpha, const FFLAS_ELT *x, const size_t incx, const FFLAS_ELT *y, const size_t incy, FFLAS_ELT *A, const size_t lda)`

*fger*: rank one update of a general matrix

- `template INST_OR_DECL void ftrsv (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS_UPLO Uplo, const FFLAS_TRANSPOSE TransA, const FFLAS_DIAG Diag, const size_t N, const FFLAS_ELT *A, const size_t lda, FFLAS_ELT *X, int incX)`

*ftrsv*: *TRI*angular System solve with Vector Computes  $X \leftarrow \text{op}(A^{-1})X$

- `template INST_OR_DECL void ftrsm (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS_SIDE Side, const FFLAS_UPLO Uplo, const FFLAS_TRANSPOSE TransA, const FFLAS_DIAG Diag, const size_t M, const size_t N, const FFLAS_ELT alpha, const FFLAS_ELT *A, const size_t lda, FFLAS_ELT *B, const size_t ldb)`

*ftrsm*: *TRI*angular System solve with *M*atrix.

- `template INST_OR_DECL void ftrmm (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS_SIDE Side, const FFLAS_UPLO Uplo, const FFLAS_TRANSPOSE TransA, const FFLAS_DIAG Diag, const size_t M, const size_t N, const FFLAS_ELT alpha, const FFLAS_ELT *A, const size_t lda, FFLAS_ELT *B, const size_t ldb)`

*ftrmm*: *TRI*angular *M*atrix *M*ultiply.

- `template INST_OR_DECL FFLAS_ELT * fgemm (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const FFLAS_ELT alpha, const FFLAS_ELT *A, const size_t lda, const FFLAS_ELT *B, const size_t ldb, const FFLAS_ELT beta, FFLAS_ELT *C, const size_t ldc)`

*fgemm*: *Field GEneral Matrix Multiply*.

- `template INST_OR_DECL FFLAS_ELT * fgemm (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const FFLAS_ELT alpha, const FFLAS_ELT *A, const size_t lda, const FFLAS_ELT *B, const size_t ldb, const FFLAS_ELT beta, FFLAS_ELT *C, const size_t ldc, const ParSeqHelper::Sequential seq)`
- `template INST_OR_DECL FFLAS_ELT * fgemm (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const FFLAS_ELT alpha, const FFLAS_ELT *A, const size_t lda, const FFLAS_ELT *B, const size_t ldb, const FFLAS_ELT beta, FFLAS_ELT *C, const size_t ldc, const ParSeqHelper::Parallel< CuttingStrategy::Recursive, StrategyParameter::TwoDAdaptive > par)`
- `template INST_OR_DECL FFLAS_ELT * fgemm (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const FFLAS_ELT alpha, const FFLAS_ELT *A, const size_t lda, const FFLAS_ELT *B, const size_t ldb, const FFLAS_ELT beta, FFLAS_ELT *C, const size_t ldc, const ParSeqHelper::Parallel< CuttingStrategy::Block, StrategyParameter::Threads > par)`
- `template INST_OR_DECL FFLAS_ELT * fsquare (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS_TRANSPOSE ta, const size_t n, const FFLAS_ELT alpha, const FFLAS_ELT *A, const size_t lda, const FFLAS_ELT beta, FFLAS_ELT *C, const size_t ldc)`

*fsquare*: Squares a matrix.

- `template<class Cut = CuttingStrategy::Block, class Strat = StrategyParameter::Threads>  
void BlockCuts (size_t &RBLOCKSIZE, size_t &CBLOCKSIZE, const size_t m, const size_t n, const size_t numthreads)`
- `template<> void BlockCuts< CuttingStrategy::Single, StrategyParameter::Threads > (size_t &RBLOCKSIZE, size_t &CBLOCKSIZE, const size_t m, const size_t n, const size_t numthreads)`

- `template<> void BlockCuts< CuttingStrategy::Row, StrategyParameter::Fixed > (size_t &RBLOCKSIZE, size_t &CBLOCKSIZE, const size_t m, const size_t n, const size_t numthreads)`
- `template<> void BlockCuts< CuttingStrategy::Row, StrategyParameter::Grain > (size_t &RBLOCKSIZE, size_t &CBLOCKSIZE, const size_t m, const size_t n, const size_t grainsize)`
- `template<> void BlockCuts< CuttingStrategy::Block, StrategyParameter::Grain > (size_t &RBLOCKSIZE, size_t &CBLOCKSIZE, const size_t m, const size_t n, const size_t grainsize)`
- `template<> void BlockCuts< CuttingStrategy::Column, StrategyParameter::Fixed > (size_t &RBLOCKSIZE, size_t &CBLOCKSIZE, const size_t m, const size_t n, const size_t numthreads)`
- `template<> void BlockCuts< CuttingStrategy::Column, StrategyParameter::Grain > (size_t &RBLOCKSIZE, size_t &CBLOCKSIZE, const size_t m, const size_t n, const size_t grainsize)`
- `template<> void BlockCuts< CuttingStrategy::Block, StrategyParameter::Fixed > (size_t &RBLOCKSIZE, size_t &CBLOCKSIZE, const size_t m, const size_t n, const size_t numthreads)`
- `template<> void BlockCuts< CuttingStrategy::Row, StrategyParameter::Threads > (size_t &RBLOCKSIZE, size_t &CBLOCKSIZE, const size_t m, const size_t n, const size_t numthreads)`
- `template<> void BlockCuts< CuttingStrategy::Column, StrategyParameter::Threads > (size_t &RBLOCKSIZE, size_t &CBLOCKSIZE, const size_t m, const size_t n, const size_t numthreads)`
- `template<> void BlockCuts< CuttingStrategy::Block, StrategyParameter::Threads > (size_t &RBLOCKSIZE, size_t &CBLOCKSIZE, const size_t m, const size_t n, const size_t numthreads)`
- `template<class Cut = CuttingStrategy::Block, class Param = StrategyParameter::Threads>  
void BlockCuts (size_t &rowBlockSize, size_t &colBlockSize, size_t &lastRBS, size_t &lastCBS, size_t &changeRBS, size_t &changeCBS, size_t &numRowBlock, size_t &numColBlock, size_t m, size_t n, const size_t numthreads)`
- `template<class Field >  
void pfzero (const Field &F, size_t m, size_t n, typename Field::Element_ptr C, size_t BS=0)`
- `template<class Field, class RandIter >  
void pfrand (const Field &F, RandIter &G, size_t m, size_t n, typename Field::Element_ptr C, size_t BS=0)`
- `template<class Field, class Cut, class Param >  
Field::Element & fdot (const Field &F, const size_t N, typename Field::ConstElement_ptr x, const size_t incx, typename Field::ConstElement_ptr y, const size_t incy, typename Field::Element &d, const ParSeqHelper::Parallel< Cut, Param > par)`
- `template<class Field, class AlgoT, class FieldTrait >  
Field::Element * pfgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, const typename Field::ConstElement_ptr A, const size_t lda, const typename Field::ConstElement_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element *C, const size_t ldc, MMHelper< Field, AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Block, StrategyParameter::Threads > > &H)`
- `template<class Field, class AlgoT, class FieldTrait >  
Field::Element * pfgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, const typename Field::ConstElement_ptr AA, const size_t lda, const typename Field::ConstElement_ptr BB, const size_t ldb, const typename Field::Element beta, typename Field::Element *C, const size_t ldc, MMHelper< Field, AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Recursive, StrategyParameter::ThreeDAdaptive > > &H)`
- `template<class Field, class AlgoT, class FieldTrait >  
Field::Element * pfgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, const typename Field::ConstElement_ptr AA, const size_t lda, const typename Field::ConstElement_ptr BB, const size_t ldb, const typename Field::Element beta, typename Field::Element *C, const size_t ldc, MMHelper< Field, AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Recursive, StrategyParameter::TwoDAdaptive > > &H)`
- `template<class Field, class AlgoT, class FieldTrait >  
Field::Element * pfgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, const typename Field::ConstElement_ptr AA, const size_t lda, const typename Field::ConstElement_ptr BB, const size_t ldb, const typename Field::Element beta, typename Field::Element *C, const size_t ldc, MMHelper< Field, AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Recursive, StrategyParameter::TwoD > > &H)`
- `template<class Field, class AlgoT, class FieldTrait >  
Field::Element_ptr pfgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE`

- tb, const size\_t m, const size\_t n, const size\_t k, const typename Field::Element alpha, const typename Field::ConstElement\_ptr A, const size\_t lda, const typename Field::ConstElement\_ptr B, const size\_t ldb, const typename Field::Element beta, typename Field::Element\_ptr C, const size\_t ldc, MMHelper< Field, AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Recursive, StrategyParameter::ThreeD > > &H)
- template<class Field, class AlgoT, class FieldTrait >  
Field::Element \* pfgemm (const Field &F, const FFLAS\_TRANSPOSE ta, const FFLAS\_TRANSPOSE tb, const size\_t m, const size\_t n, const size\_t k, const typename Field::Element alpha, const typename Field::ConstElement\_ptr A, const size\_t lda, const typename Field::ConstElement\_ptr B, const size\_t ldb, const typename Field::Element beta, typename Field::Element\_ptr C, const size\_t ldc, MMHelper< Field, AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Recursive, StrategyParameter::ThreeDInPlace > > &H)
- template<class Field, class AlgoT, class FieldTrait >  
Field::Element\_ptr fgemv (const Field &F, const FFLAS\_TRANSPOSE ta, const size\_t m, const size\_t n, const typename Field::Element alpha, const typename Field::ConstElement\_ptr A, const size\_t lda, const typename Field::ConstElement\_ptr X, const size\_t incX, const typename Field::Element beta, typename Field::Element\_ptr Y, const size\_t incY, MMHelper< Field, AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Recursive, StrategyParameter::Threads > > &H)
- template<class Field, class AlgoT, class FieldTrait, class Cut >  
Field::Element\_ptr fgemv (const Field &F, const FFLAS\_TRANSPOSE ta, const size\_t m, const size\_t n, const typename Field::Element alpha, const typename Field::ConstElement\_ptr A, const size\_t lda, const typename Field::ConstElement\_ptr X, const size\_t incX, const typename Field::Element beta, typename Field::Element\_ptr Y, const size\_t incY, MMHelper< Field, AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Row, Cut > > &H)
- void parseArguments (int argc, char \*\*argv, Argument \*args, bool printDefaults=true)
- char \* getArgumentsValue (int argc, char \*\*argv, int i)  
*Get the value of an argument and avoid core dump when no value was given after an argument.*
- std::ostream & writeCommandString (std::ostream &os, Argument \*args, const char \*programName=nullptr)  
*writes the values of all arguments, preceded by the programName*
- template<class Field >  
std::ostream & WriteMatrix (std::ostream &c, const Field &F, size\_t m, size\_t n, typename Field::ConstElement\_ptr A, size\_t lda, FFLAS\_FORMAT format, bool column\_major)  
*WriteMatrix: write a matrix to an output stream.*
- void preamble (std::ifstream &ifs, FFLAS\_FORMAT &format)
- template<class Field >  
Field::Element\_ptr ReadMatrix (std::ifstream &ifs, Field &F, size\_t &m, size\_t &n, typename Field::Element\_ptr &A, FFLAS\_FORMAT format=FflasAuto)  
*ReadMatrix: read a matrix from an input stream.*
- template<class Field >  
Field::Element\_ptr ReadMatrix (const std::string &matrix\_file, Field &F, size\_t &m, size\_t &n, typename Field::Element\_ptr &A, FFLAS\_FORMAT format=FflasAuto)  
*ReadMatrix: read a matrix from a file.*
- template<class Field >  
void WriteMatrix (std::string &matrix\_file, const Field &F, int m, int n, typename Field::ConstElement\_ptr A, size\_t lda, FFLAS\_FORMAT format=FflasDense, bool column\_major=false)  
*WriteMatrix: write a matrix to a file.*
- std::ostream & WritePermutation (std::ostream &c, const size\_t \*P, size\_t N)  
*WritePermutation: write a permutation matrix to an output stream.*
- template<class Element >  
bool alignable ()
- template<> bool alignable< Givaro::Integer \* > ()
- template<class Field >  
Field::Element\_ptr fflas\_new (const Field &F, const size\_t m, const Alignment align=Alignment::DEFAULT)
- template<class Field >  
Field::Element\_ptr fflas\_new (const Field &F, const size\_t m, const size\_t n, const Alignment align=Alignment::DEFAULT)

- `template<class Element >`  
`Element * fflas_new (const size_t m, const Alignment align=Alignment::DEFAULT)`
- `template<class Element_ptr >`  
`void fflas_delete (Element_ptr A)`
- `template<class Ptr , class ... Args>`  
`void fflas_delete (Ptr p, Args ... args)`
- `void prefetch (const int64_t *)`
- `void getTLBSize (int &tlb)`
- `void queryCacheSizes (int &l1, int &l2, int &l3)`
- `int queryL1CacheSize ()`
- `int queryTopLevelCacheSize ()`
- `uint64_t getSeed ()`

### 11.1.1 Typedef Documentation

#### 11.1.1.1 Checker\_fgemm

```
template<class Field >
using Checker_fgemm = FFLAS::Checker_Empty<Field>
```

#### 11.1.1.2 Checker\_ftrsm

```
template<class Field >
using Checker_ftrsm = FFLAS::Checker_Empty<Field>
```

#### 11.1.1.3 ForceCheck\_fgemm

```
template<class Field >
using ForceCheck_fgemm = CheckerImplem_fgemm<Field>
```

#### 11.1.1.4 ForceCheck\_ftrsm

```
template<class Field >
using ForceCheck_ftrsm = CheckerImplem_ftrsm<Field>
```

#### 11.1.1.5 ZOSparseMatrix

```
using ZOSparseMatrix = std::true_type
```

#### 11.1.1.6 NotZOSparseMatrix

```
using NotZOSparseMatrix = std::false_type
```

#### 11.1.1.7 SimdSparseMatrix

```
using SimdSparseMatrix = std::true_type
```

#### 11.1.1.8 NoSimdSparseMatrix

```
using NoSimdSparseMatrix = std::false_type
```

#### 11.1.1.9 MKLSparseMatrixFormat

```
using MKLSparseMatrixFormat = std::true_type
```

#### 11.1.1.10 NotMKLSparseMatrixFormat

```
using NotMKLSparseMatrixFormat = std::false_type
```

#### 11.1.1.11 has\_plus

```
template<class T >
using has_plus = typename std::conditional<std::is_arithmetic<T>::value, std::true_type,
has_plus_impl<T> >::type
```

#### 11.1.1.12 has\_minus

```
template<class T >
using has_minus = typename std::conditional<std::is_arithmetic<T>::value, std::true_type,
has_minus_impl<T> >::type
```

#### 11.1.1.13 has\_equal

```
template<class T >
using has_equal = typename std::conditional<std::is_arithmetic<T>::value, std::true_type,
std::is_copy_assignable<T> >::type
```

#### 11.1.1.14 has\_plus\_eq

```
template<class T >
using has_plus_eq = typename std::conditional<std::is_arithmetic<T>::value, std::true_type,
has_plus_eq_impl<T> >::type
```

#### 11.1.1.15 has\_minus\_eq

```
template<class T >
using has_minus_eq = typename std::conditional<std::is_arithmetic<T>::value, std::true_type,
has_minus_eq_impl<T> >::type
```

**11.1.1.16 has\_mul**

```
template<class T >
using has_mul = typename std::conditional<std::is_arithmetic<T>::value, std::true_type, has_mul_impl<T>
>::type
```

**11.1.1.17 has\_mul\_eq**

```
template<class T >
using has_mul_eq = typename std::conditional<std::is_arithmetic<T>::value, std::true_type,
has_mul_eq_impl<T> >::type
```

**11.1.1.18 Timer**

```
typedef Givaro::Timer Timer
```

**11.1.1.19 BaseTimer**

```
typedef Givaro::BaseTimer BaseTimer
```

**11.1.1.20 UserTimer**

```
typedef Givaro::UserTimer UserTimer
```

**11.1.1.21 SysTimer**

```
typedef Givaro::SysTimer SysTimer
```

**11.1.2 Enumeration Type Documentation****11.1.2.1 FFLAS\_ORDER**

```
enum FFLAS_ORDER
```

Storage by row or col ?

Enumerator

|               |           |
|---------------|-----------|
| FflasRowMajor | row major |
| FflasColMajor | col major |



## 11.1.2.2 FFLAS\_TRANSPOSE

```
enum FFLAS_TRANSPOSE
```

Is matrix transposed ?

Enumerator

|              |                           |
|--------------|---------------------------|
| FflasNoTrans | Matrix is not transposed. |
| FflasTrans   | Matrix is transposed.     |

## 11.1.2.3 FFLAS\_UPLO

```
enum FFLAS_UPLO
```

Is triangular matrix's shape upper ?

Enumerator

|               |                                                                                |
|---------------|--------------------------------------------------------------------------------|
| FflasUpper    | Triangular matrix is Upper triangular (if $i > j$ then $T_{i,j} = 0$ )         |
| FflasLower    | Triangular matrix is Lower triangular (if $i < j$ then $T_{i,j} = 0$ )         |
| FflasLeftTri  | Triangular matrix is Left triangular (if $j > n - i - 1$ then $T_{i,j} = 0$ )  |
| FflasRightTri | Triangular matrix is Right triangular (if $j < n - i - 1$ then $T_{i,j} = 0$ ) |

## 11.1.2.4 FFLAS\_DIAG

```
enum FFLAS_DIAG
```

Is the triangular matrix implicitly unit diagonal ?

Enumerator

|              |                                                                   |
|--------------|-------------------------------------------------------------------|
| FflasNonUnit | Triangular matrix has an explicit arbitrary diagonal.             |
| FflasUnit    | Triangular matrix has an implicit unit diagonal ( $T_{i,i} = 1$ ) |

## 11.1.2.5 FFLAS\_SIDE

```
enum FFLAS_SIDE
```

On what side ?

Enumerator

|            |                                 |
|------------|---------------------------------|
| FflasLeft  | Operator applied on the left.   |
| FflasRight | Operator applied on the righth. |

### 11.1.2.6 FFLAS\_BASE

`enum FFLAS_BASE`

FFLAS\_BASE determines the type of the element representation for Matrix Mult kernel.

(deprecated, should not be used)

#### Enumerator

|              |                                                                            |
|--------------|----------------------------------------------------------------------------|
| FflasDouble  | to use the double precision BLAS                                           |
| FflasFloat   | to use the single precison BLAS                                            |
| FflasGeneric | for any other domain, that can not be converted to floating point integers |

### 11.1.2.7 number\_kind

`enum number_kind`

#### Enumerator

|       |  |
|-------|--|
| zero  |  |
| one   |  |
| mone  |  |
| other |  |

### 11.1.2.8 SparseMatrix\_t

`enum class SparseMatrix_t [strong]`

#### Enumerator

|             |  |
|-------------|--|
| CSR         |  |
| CSR_ZO      |  |
| CSC         |  |
| CSC_ZO      |  |
| COO         |  |
| COO_ZO      |  |
| ELL         |  |
| ELL_ZO      |  |
| SELL        |  |
| SELL_ZO     |  |
| ELL_simd    |  |
| ELL_simd_ZO |  |
| CSR_HYB     |  |
| HYB_ZO      |  |

### 11.1.2.9 FFLAS\_FORMAT

enum FFLAS\_FORMAT

Enumerator

|               |  |
|---------------|--|
| FflasAuto     |  |
| FflasDense    |  |
| FflasSMS      |  |
| FflasBinary   |  |
| FflasMath     |  |
| FflasMaple    |  |
| FflasSageMath |  |

## 11.1.3 Function Documentation

### 11.1.3.1 InfNorm()

```
Givaro::Integer InfNorm (
 const size_t M,
 const size_t N,
 const Givaro::Integer * A,
 const size_t lda) [inline]
```

### 11.1.3.2 min3()

```
template<class T >
const T & min3 (
 const T & m,
 const T & n,
 const T & k)
```

### 11.1.3.3 max3()

```
template<class T >
const T & max3 (
 const T & m,
 const T & n,
 const T & k)
```

### 11.1.3.4 min4()

```
template<class T >
const T & min4 (
 const T & m,
 const T & n,
 const T & k,
 const T & l)
```

**11.1.3.5 max4()**

```
template<class T >
const T & max4 (
 const T & m,
 const T & n,
 const T & k,
 const T & l)
```

**11.1.3.6 fadd() [1/8]**

```
template<class Field >
void fadd (
 const Field & F,
 const size_t N,
 typename Field::ConstElement_ptr A,
 const size_t inca,
 typename Field::ConstElement_ptr B,
 const size_t incb,
 typename Field::Element_ptr C,
 const size_t incc)
```

**11.1.3.7 faddin() [1/5]**

```
template<class Field >
void faddin (
 const Field & F,
 const size_t N,
 typename Field::ConstElement_ptr B,
 const size_t incb,
 typename Field::Element_ptr C,
 const size_t incc)
```

**11.1.3.8 fsub() [1/4]**

```
template<class Field >
void fsub (
 const Field & F,
 const size_t N,
 typename Field::ConstElement_ptr A,
 const size_t inca,
 typename Field::ConstElement_ptr B,
 const size_t incb,
 typename Field::Element_ptr C,
 const size_t incc)
```

**11.1.3.9 fsubin() [1/3]**

```
template<class Field >
void fsubin (
 const Field & F,
 const size_t N,
 typename Field::ConstElement_ptr B,
 const size_t incb,
 typename Field::Element_ptr C,
 const size_t incc)
```

**11.1.3.10 fadd()** [2/8]

```

template<class Field >
void fadd (
 const Field & F,
 const size_t N,
 typename Field::ConstElement_ptr A,
 const size_t inca,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr B,
 const size_t incb,
 typename Field::Element_ptr C,
 const size_t incc)

```

**Todo** optimise here

**11.1.3.11 pfadd()**

```

template<class Field >
void pfadd (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr B,
 const size_t ldb,
 typename Field::Element_ptr C,
 const size_t ldc,
 const size_t numths)

```

**11.1.3.12 pfsb()**

```

template<class Field >
void pfsb (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr B,
 const size_t ldb,
 typename Field::Element_ptr C,
 const size_t ldc,
 const size_t numths)

```

### 11.1.3.13 pfaddin()

```
template<class Field >
void pfaddin (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::ConstElement_ptr B,
 const size_t ldb,
 typename Field::Element_ptr C,
 const size_t ldc,
 size_t numths)
```

### 11.1.3.14 pfsubin()

```
template<class Field >
void pfsubin (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::ConstElement_ptr B,
 const size_t ldb,
 typename Field::Element_ptr C,
 const size_t ldc,
 size_t numths)
```

### 11.1.3.15 fadd() [3/8]

```
template<class Field >
void fadd (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr B,
 const size_t ldb,
 typename Field::Element_ptr C,
 const size_t ldc)
```

fadd : matrix addition.

Computes  $C = A + B$ .

#### Parameters

|            |                          |
|------------|--------------------------|
| <i>F</i>   | field                    |
| <i>M</i>   | rows                     |
| <i>N</i>   | cols                     |
| <i>A</i>   | dense matrix of size MxN |
| <i>lda</i> | leading dimension of A   |
| <i>B</i>   | dense matrix of size MxN |
| <i>ldb</i> | leading dimension of B   |
| <i>C</i>   | dense matrix of size MxN |
| <i>ldc</i> | leading dimension of C   |

**11.1.3.16 fsub()** [2/4]

```

template<class Field >
void fsub (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr B,
 const size_t ldb,
 typename Field::Element_ptr C,
 const size_t ldc)

```

fsub : matrix subtraction.

Computes  $C = A - B$ .

**Parameters**

|            |                          |
|------------|--------------------------|
| <i>F</i>   | field                    |
| <i>M</i>   | rows                     |
| <i>N</i>   | cols                     |
| <i>A</i>   | dense matrix of size MxN |
| <i>lda</i> | leading dimension of A   |
| <i>B</i>   | dense matrix of size MxN |
| <i>ldb</i> | leading dimension of B   |
| <i>C</i>   | dense matrix of size MxN |
| <i>ldc</i> | leading dimension of C   |

**11.1.3.17 faddin()** [2/5]

```

template<class Field >
void faddin (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::ConstElement_ptr B,
 const size_t ldb,
 typename Field::Element_ptr C,
 const size_t ldc)

```

faddin

**11.1.3.18 faddin()** [3/5]

```

template<class Field >
void faddin (
 const Field & F,
 const FFLAS_UPLO uplo,
 const size_t N,

```

```

 typename Field::ConstElement_ptr B,
 const size_t ldb,
 typename Field::Element_ptr C,
 const size_t ldc)

```

fadding for symmetric matrices

#### 11.1.3.19 fsubin() [2/3]

```

template<class Field >
void fsubin (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::ConstElement_ptr B,
 const size_t ldb,
 typename Field::Element_ptr C,
 const size_t ldc)

```

fsubin  $C = C - B$

#### 11.1.3.20 fadd() [4/8]

```

template<class Field >
void fadd (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr B,
 const size_t ldb,
 typename Field::Element_ptr C,
 const size_t ldc)

```

fadd : matrix addition with scaling.

Computes  $C = A + \text{alpha } B$ .

##### Parameters

|              |                                   |
|--------------|-----------------------------------|
| <i>F</i>     | field                             |
| <i>M</i>     | rows                              |
| <i>N</i>     | cols                              |
| <i>A</i>     | dense matrix of size $M \times N$ |
| <i>lda</i>   | leading dimension of A            |
| <i>alpha</i> | some scalar                       |
| <i>B</i>     | dense matrix of size $M \times N$ |
| <i>ldb</i>   | leading dimension of B            |
| <i>C</i>     | dense matrix of size $M \times N$ |
| <i>ldc</i>   | leading dimension of C            |



**11.1.3.21 fassign()** [1/10]

```
template<class Field >
void fassign (
 const Field & F,
 const size_t N,
 typename Field::ConstElement_ptr Y,
 const size_t incY,
 typename Field::Element_ptr X,
 const size_t incX) [inline]
```

fassign :  $x \leftarrow y$ .

X is preallocated

**Todo** variant for triangular matrix

**Parameters**

|     |        |                     |
|-----|--------|---------------------|
|     | $F$    | field               |
|     | $N$    | size of the vectors |
| out | $X$    | vector in F         |
|     | $incX$ | stride of X         |
| in  | $Y$    | vector in F         |
|     | $incY$ | stride of Y         |

**11.1.3.22 fassign()** [2/10]

```
template<>
void fassign (
 const Givaro::Modular< float > & F,
 const size_t N,
 const float * Y,
 const size_t incY,
 float * X,
 const size_t incX) [inline]
```

**11.1.3.23 fassign()** [3/10]

```
template<>
void fassign (
 const Givaro::ModularBalanced< float > & F,
 const size_t N,
 const float * Y,
 const size_t incY,
 float * X,
 const size_t incX) [inline]
```

**11.1.3.24 fassign()** [4/10]

```
template<>
void fassign (
 const Givaro::ZRing< float > & F,
 const size_t N,
 const float * Y,
 const size_t incY,
 float * X,
 const size_t incX) [inline]
```

**11.1.3.25 fassign()** [5/10]

```
template<>
void fassign (
 const Givaro::Modular< double > & F,
 const size_t N,
 const double * Y,
 const size_t incY,
 double * X,
 const size_t incX) [inline]
```

**11.1.3.26 fassign()** [6/10]

```
template<>
void fassign (
 const Givaro::ModularBalanced< double > & F,
 const size_t N,
 const double * Y,
 const size_t incY,
 double * X,
 const size_t incX) [inline]
```

**11.1.3.27 fassign()** [7/10]

```
template<>
void fassign (
 const Givaro::ZRing< double > & F,
 const size_t N,
 const double * Y,
 const size_t incY,
 double * X,
 const size_t incX) [inline]
```

**11.1.3.28 fassign()** [8/10]

```
template<class Field >
void fassign (
 const Field & F,
 const size_t m,
 const size_t n,
 typename Field::ConstElement_ptr B,
 const size_t ldb,
 typename Field::Element_ptr A,
 const size_t lda)
```

fassign :  $A \leftarrow B$ .

## Parameters

|       |                        |
|-------|------------------------|
| $F$   | field                  |
| $m$   | number of rows to copy |
| $n$   | number of cols to copy |
| $A$   | matrix in $F$          |
| $lda$ | stride of $A$          |
| $B$   | vector in $F$          |
| $ldb$ | stride of $B$          |

## 11.1.3.29 faxpy() [1/6]

```
template<class Field >
void faxpy (
 const Field & F,
 const size_t N,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr X,
 const size_t incX,
 typename Field::Element_ptr Y,
 const size_t incY) [inline]
```

$\text{faxpy} : y \leftarrow \alpha \cdot x + y.$

## Parameters

|         |               |                     |
|---------|---------------|---------------------|
|         | $F$           | field               |
|         | $N$           | size of the vectors |
|         | $\alpha$      | scalar              |
| in      | $X$           | vector in $F$       |
|         | $\text{incX}$ | stride of $X$       |
| in, out | $Y$           | vector in $F$       |
|         | $\text{incY}$ | stride of $Y$       |

## 11.1.3.30 faxpy() [2/6]

```
template<>
void faxpy (
 const Givaro::DoubleDomain & ,
 const size_t N,
 const Givaro::DoubleDomain::Element a,
 Givaro::DoubleDomain::ConstElement_ptr x,
 const size_t incx,
 Givaro::DoubleDomain::Element_ptr y,
 const size_t incy) [inline]
```

## 11.1.3.31 faxpy() [3/6]

```
template<>
void faxpy (
```

```

const Givaro::FloatDomain & ,
const size_t N,
const Givaro::FloatDomain::Element a,
Givaro::FloatDomain::ConstElement_ptr x,
const size_t incx,
Givaro::FloatDomain::Element_ptr y,
const size_t incy) [inline]

```

### 11.1.3.32 faxpy() [4/6]

```

template<class Field >
void faxpy (
 const Field & F,
 const size_t m,
 const size_t n,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr X,
 const size_t ldx,
 typename Field::Element_ptr Y,
 const size_t ldy) [inline]

```

$\text{faxpy} : y \leftarrow \alpha \cdot x + y.$

#### Parameters

|         |          |                          |
|---------|----------|--------------------------|
|         | $F$      | field                    |
|         | $m$      | row dimension            |
|         | $n$      | column dimension         |
|         | $\alpha$ | scalar                   |
| in      | $X$      | vector in $F$            |
|         | $ldx$    | leading dimension of $X$ |
| in, out | $Y$      | vector in $F$            |
|         | $ldy$    | leading dimension of $Y$ |

### 11.1.3.33 fdot() [1/11]

```

template<class Field >
Field::Element fdot (
 const Field & F,
 const size_t N,
 typename Field::ConstElement_ptr x,
 const size_t incx,
 typename Field::ConstElement_ptr y,
 const size_t incy,
 ModeCategories::DefaultTag & MT) [inline]

```

### 11.1.3.34 fdot() [2/11]

```

template<class Field >
Field::Element fdot (

```

```

 const Field & F,
 const size_t N,
 typename Field::ConstElement_ptr x,
 const size_t incx,
 typename Field::ConstElement_ptr y,
 const size_t incy,
 ModeCategories::DelayedTag & MT) [inline]

```

#### 11.1.3.35 fdot() [3/11]

```

template<>
Givaro::DoubleDomain::Element fdot (
 const Givaro::DoubleDomain & ,
 const size_t N,
 Givaro::DoubleDomain::ConstElement_ptr x,
 const size_t incx,
 Givaro::DoubleDomain::ConstElement_ptr y,
 const size_t incy,
 ModeCategories::DefaultTag & MT) [inline]

```

#### 11.1.3.36 fdot() [4/11]

```

template<>
Givaro::FloatDomain::Element fdot (
 const Givaro::FloatDomain & ,
 const size_t N,
 Givaro::FloatDomain::ConstElement_ptr x,
 const size_t incx,
 Givaro::FloatDomain::ConstElement_ptr y,
 const size_t incy,
 ModeCategories::DefaultTag & MT) [inline]

```

#### 11.1.3.37 fdot() [5/11]

```

template<class Field , class T >
Field::Element fdot (
 const Field & F,
 const size_t N,
 typename Field::ConstElement_ptr x,
 const size_t incx,
 typename Field::ConstElement_ptr y,
 const size_t incy,
 ModeCategories::ConvertTo< T > & MT) [inline]

```

#### 11.1.3.38 fdot() [6/11]

```

template<class Field >
Field::Element fdot (
 const Field & F,
 const size_t N,
 typename Field::ConstElement_ptr x,
 const size_t incx,
 typename Field::ConstElement_ptr y,
 const size_t incy,
 ModeCategories::DefaultBoundedTag & dbt) [inline]

```

**11.1.3.39 fdot()** [7/11]

```
template<class Field >
Field::Element fdot (
 const Field & F,
 const size_t N,
 typename Field::ConstElement_ptr x,
 const size_t incx,
 typename Field::ConstElement_ptr y,
 const size_t incy,
 const ParSeqHelper::Sequential seq) [inline]
```

**11.1.3.40 fdot()** [8/11]

```
template<class Field >
Field::Element fdot (
 const Field & F,
 const size_t N,
 typename Field::ConstElement_ptr X,
 const size_t incX,
 typename Field::ConstElement_ptr Y,
 const size_t incY) [inline]
```

fdot: dot product  $x^T y$ .

**Parameters**

|        |                     |
|--------|---------------------|
| $F$    | field               |
| $N$    | size of the vectors |
| $X$    | vector in $F$       |
| $incX$ | stride of $X$       |
| $Y$    | vector in $F$       |
| $incY$ | stride of $Y$       |

**11.1.3.41 fgemm()** [1/23]

```
template<class Field >
Field::Element_ptr fgemm (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr B,
 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
```

```

 MMHelper< Field, MMHelperAlgo::Winograd, ModeCategories::ConvertTo< ElementCategories::MachineFl
>, ParSeqHelper::Sequential > & H) [inline]

```

#### 11.1.3.42 fgemm() [2/23]

```

template<typename Field >
Field::Element_ptr fgemm (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr B,
 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 const ParSeqHelper::Sequential seq) [inline]

```

#### 11.1.3.43 fgemm() [3/23]

```

template<typename Field , class Cut , class Param >
Field::Element_ptr fgemm (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr B,
 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 const ParSeqHelper::Parallel< Cut, Param > par) [inline]

```

#### 11.1.3.44 fgemm() [4/23]

```

template<typename Field >
Field::Element_ptr fgemm (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,

```



```

 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr B,
 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc) [inline]

```

**fgemm**: Field **G**eneral **M**atrix **M**ultiply.

Computes  $C = \alpha \text{op}(A) \times \text{op}(B) + \beta C$  Automatically set Winograd recursion level

#### Parameters

|              |                                                                                                       |
|--------------|-------------------------------------------------------------------------------------------------------|
| <i>F</i>     | field.                                                                                                |
| <i>ta</i>    | if $ta == \text{FflasTrans}$ then $\text{op}(A) = A^t$ , else $\text{op}(A) = A$ ,                    |
| <i>tb</i>    | same for matrix B                                                                                     |
| <i>m</i>     | see A                                                                                                 |
| <i>n</i>     | see B                                                                                                 |
| <i>k</i>     | see A                                                                                                 |
| <i>alpha</i> | scalar                                                                                                |
| <i>beta</i>  | scalar                                                                                                |
| <i>A</i>     | $\text{op}(A)$ is $m \times k$                                                                        |
| <i>B</i>     | $\text{op}(B)$ is $k \times n$                                                                        |
| <i>C</i>     | $C$ is $m \times n$                                                                                   |
| <i>lda</i>   | leading dimension of A                                                                                |
| <i>ldb</i>   | leading dimension of B                                                                                |
| <i>ldc</i>   | leading dimension of C                                                                                |
| <i>w</i>     | recursive levels of Winograd's algorithm are used. No argument (or -1) does auto computation of $w$ . |

#### Warning

$\alpha$  must be invertible

#### 11.1.3.45 fgemm() [5/23]

```

template<typename Field , class ModeT , class ParSeq >
Field::Element_ptr fgemm (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr B,
 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 MMHelper< Field, MMHelperAlgo::Auto, ModeT, ParSeq > & H) [inline]

```

### 11.1.3.46 fgemm() [6/23]

```
template<class Field >
Field::Element_ptr fgemm (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr B,
 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 MMHelper< Field, MMHelperAlgo::Winograd, ModeCategories::DelayedTag, ParSeqHelper::Sequential
> & H) [inline]
```

### 11.1.3.47 fsquare() [1/6]

```
template<class Field >
Field::Element_ptr fsquare (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const size_t n,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc) [inline]
```

fsquare: Squares a matrix.

compute  $C \leftarrow \alpha \text{op}(A) \text{op}(A) + \beta C$  over a Field  $F$  Avoid the conversion of  $B$

#### Parameters

|              |                                                       |
|--------------|-------------------------------------------------------|
| <i>ta</i>    | if $ta == \text{FflasTrans}$ , $\text{op}(A) = A^T$ . |
| <i>F</i>     | field                                                 |
| <i>n</i>     | size of A                                             |
| <i>alpha</i> | scalar                                                |
| <i>beta</i>  | scalar                                                |
| <i>A</i>     | dense matrix of size $n \times n$                     |
| <i>lda</i>   | leading dimension of A                                |
| <i>C</i>     | dense matrix of size $n \times n$                     |
| <i>ldc</i>   | leading dimension of C                                |

**Bug** why double ?

**11.1.3.48 fsquare()** [2/6]

```
template<>
double * fsquare (
 const Givaro::ModularBalanced< double > & F,
 const FFLAS_TRANSPOSE ta,
 const size_t n,
 const double alpha,
 const double * A,
 const size_t lda,
 const double beta,
 double * C,
 const size_t ldc) [inline]
```

**11.1.3.49 fsquare()** [3/6]

```
template<>
float * fsquare (
 const Givaro::ModularBalanced< float > & F,
 const FFLAS_TRANSPOSE ta,
 const size_t n,
 const float alpha,
 const float * A,
 const size_t lda,
 const float beta,
 float * C,
 const size_t ldc) [inline]
```

**11.1.3.50 fsquare()** [4/6]

```
template<>
double * fsquare (
 const Givaro::Modular< double > & F,
 const FFLAS_TRANSPOSE ta,
 const size_t n,
 const double alpha,
 const double * A,
 const size_t lda,
 const double beta,
 double * C,
 const size_t ldc) [inline]
```

**11.1.3.51 fsquare()** [5/6]

```
template<>
float * fsquare (
 const Givaro::Modular< float > & F,
 const FFLAS_TRANSPOSE ta,
 const size_t n,
 const float alpha,
 const float * A,
 const size_t lda,
 const float beta,
 float * C,
 const size_t ldc) [inline]
```

**11.1.3.52 fgemm()** [7/23]

```

template<typename RNS , typename ParSeqTrait >
FFPACK::RNSInteger< RNS >::Element_ptr fgemm (
 const FFPACK::RNSInteger< RNS > & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const typename FFPACK::RNSInteger< RNS >::Element alpha,
 typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Ad,
 const size_t lda,
 typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Bd,
 const size_t ldb,
 const typename FFPACK::RNSInteger< RNS >::Element beta,
 typename FFPACK::RNSInteger< RNS >::Element_ptr Cd,
 const size_t ldc,
 MMHelper< FFPACK::RNSInteger< RNS >, MMHelperAlgo::Classic, ModeCategories::DefaultTag,
 ParSeqHelper::Compose< ParSeqHelper::Sequential, ParSeqTrait > > & H) [inline]

```

**11.1.3.53 fgemm()** [8/23]

```

template<typename RNS >
FFPACK::RNSInteger< RNS >::Element_ptr fgemm (
 const FFPACK::RNSInteger< RNS > & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const typename FFPACK::RNSInteger< RNS >::Element alpha,
 typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Ad,
 const size_t lda,
 typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Bd,
 const size_t ldb,
 const typename FFPACK::RNSInteger< RNS >::Element beta,
 typename FFPACK::RNSInteger< RNS >::Element_ptr Cd,
 const size_t ldc,
 MMHelper< FFPACK::RNSInteger< RNS >, MMHelperAlgo::Classic, ModeCategories::DefaultTag,
 ParSeqHelper::Sequential > & H) [inline]

```

**11.1.3.54 fgemm()** [9/23]

```

template<typename RNS , typename ParSeqTrait >
FFPACK::RNSInteger< RNS >::Element_ptr fgemm (
 const FFPACK::RNSInteger< RNS > & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const typename FFPACK::RNSInteger< RNS >::Element alpha,
 typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Ad,

```

```

 const size_t lda,
 typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Bd,
 const size_t ldb,
 const typename FFPACK::RNSInteger< RNS >::Element beta,
 typename FFPACK::RNSInteger< RNS >::Element_ptr Cd,
 const size_t ldc,
 MMHelper< FFPACK::RNSInteger< RNS >, MMHelperAlgo::Classic, ModeCategories::DefaultTag,
 ParSeqHelper::Compose< ParSeqHelper::Parallel< CuttingStrategy::RNSModulus, StrategyParameter::Threads
 >, ParSeqTrait > > & H) [inline]

```

### 11.1.3.55 fgemm() [10/23]

```

template<typename RNS , typename Cut , typename Param >
FFPACK::RNSInteger< RNS >::Element_ptr fgemm (
 const FFPACK::RNSInteger< RNS > & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const typename FFPACK::RNSInteger< RNS >::Element alpha,
 typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Ad,
 const size_t lda,
 typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Bd,
 const size_t ldb,
 const typename FFPACK::RNSInteger< RNS >::Element beta,
 typename FFPACK::RNSInteger< RNS >::Element_ptr Cd,
 const size_t ldc,
 MMHelper< FFPACK::RNSInteger< RNS >, MMHelperAlgo::Classic, ModeCategories::DefaultTag,
 ParSeqHelper::Parallel< Cut, Param > > & H) [inline]

```

### 11.1.3.56 fgemm() [11/23]

```

template<class ParSeq >
Givaro::Integer * fgemm (
 const Givaro::ZRing< Givaro::Integer > & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const Givaro::Integer alpha,
 const Givaro::Integer * A,
 const size_t lda,
 const Givaro::Integer * B,
 const size_t ldb,
 Givaro::Integer beta,
 Givaro::Integer * C,
 const size_t ldc,
 MMHelper< Givaro::ZRing< Givaro::Integer >, MMHelperAlgo::Classic, ModeCategories::ConvertTo<
 ElementCategories::RNSElementTag >, ParSeq > & H) [inline]

```

**11.1.3.57 fgemm()** [12/23]

```

template<typename RNS , class ModeT >
RNS::Element_ptr fgemm (
 const FFPACK::RNSInteger< RNS > & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const typename RNS::Element alpha,
 typename RNS::ConstElement_ptr Ad,
 const size_t lda,
 typename RNS::ConstElement_ptr Bd,
 const size_t ldb,
 const typename RNS::Element beta,
 typename RNS::Element_ptr Cd,
 const size_t ldc,
 MMHelper< FFPACK::RNSInteger< RNS >, MMHelperAlgo::Winograd, ModeT, ParSeqHelper::Sequential
> & H) [inline]

```

**11.1.3.58 fgemm()** [13/23]

```

template<typename RNS >
RNS::Element_ptr fgemm (
 const FFPACK::RNSIntegerMod< RNS > & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const typename RNS::Element alpha,
 typename RNS::ConstElement_ptr Ad,
 const size_t lda,
 typename RNS::ConstElement_ptr Bd,
 const size_t ldb,
 const typename RNS::Element beta,
 typename RNS::Element_ptr Cd,
 const size_t ldc,
 MMHelper< FFPACK::RNSIntegerMod< RNS >, MMHelperAlgo::Winograd > & H) [inline]

```

**11.1.3.59 fgemm()** [14/23]

```

Givaro::Integer * fgemm (
 const Givaro::Modular< Givaro::Integer > & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const Givaro::Integer alpha,
 const Givaro::Integer * A,
 const size_t lda,
 const Givaro::Integer * B,

```

```

 const size_t ldb,
 const Givaro::Integer beta,
 Givaro::Integer * C,
 const size_t ldc,
 MMHelper< Givaro::Modular< Givaro::Integer >, MMHelperAlgo::Classic, ModeCategories::ConvertTo<
ElementCategories::RNSElementTag > > & H) [inline]

```

### 11.1.3.60 fgemm() [15/23]

```

template<class ParSeq >
Givaro::Integer * fgemm (
 const Givaro::Modular< Givaro::Integer > & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const Givaro::Integer alpha,
 const Givaro::Integer * A,
 const size_t lda,
 const Givaro::Integer * B,
 const size_t ldb,
 const Givaro::Integer beta,
 Givaro::Integer * C,
 const size_t ldc,
 MMHelper< Givaro::Modular< Givaro::Integer >, MMHelperAlgo::Auto, ModeCategories::ConvertTo<
ElementCategories::RNSElementTag >, ParSeq > & H) [inline]

```

### 11.1.3.61 fgemm() [16/23]

```

template<size_t K1, size_t K2, class ParSeq >
RecInt::ruint< K1 > * fgemm (
 const Givaro::Modular< RecInt::ruint< K1 >, RecInt::ruint< K2 > > & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const RecInt::ruint< K1 > alpha,
 const RecInt::ruint< K1 > * A,
 const size_t lda,
 const RecInt::ruint< K1 > * B,
 const size_t ldb,
 RecInt::ruint< K1 > beta,
 RecInt::ruint< K1 > * C,
 const size_t ldc,
 MMHelper< Givaro::Modular< RecInt::ruint< K1 >, RecInt::ruint< K2 > >, MMHelperAlgo::Classic,
ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeq > & H) [inline]

```

### 11.1.3.62 fgemm() [17/23]

```

template<class Field , class ModeT >
Field::Element_ptr fgemm (

```

```

 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr B,
 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 MMHelper< Field, MMHelperAlgo::Winograd, ModeT > & H) [inline]

```

### 11.1.3.63 fgemm() [18/23]

```

template<class Field , class ModeT , class Cut , class Param >
Field::Element_ptr fgemm (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr B,
 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 MMHelper< Field, MMHelperAlgo::WinogradPar, ModeT, ParSeqHelper::Parallel< Cut,
Param > > & H) [inline]

```

### 11.1.3.64 fgemv() [1/19]

```

template<class Field >
Field::Element_ptr fgemv (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const size_t M,
 const size_t N,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr X,
 const size_t incX,
 const typename Field::Element beta,
 typename Field::Element_ptr Y,
 const size_t incY,
 MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::ConvertTo< ElementCategories::MachineFlo
> > & H) [inline]

```



**11.1.3.65 fgemv()** [2/19]

```
template<class Field >
Field::Element_ptr fgemv (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const size_t M,
 const size_t N,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr X,
 const size_t incX,
 const typename Field::Element beta,
 typename Field::Element_ptr Y,
 const size_t incY,
 MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::DelayedTag > & H) [inline]
```

**11.1.3.66 fgemv()** [3/19]

```
template<class Field >
Field::Element_ptr fgemv (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const size_t M,
 const size_t N,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr X,
 const size_t incX,
 const typename Field::Element beta,
 typename Field::Element_ptr Y,
 const size_t incY,
 MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::DefaultTag > & H) [inline]
```

**11.1.3.67 fgemv()** [4/19]

```
template<class Field >
Field::Element_ptr fgemv (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const size_t M,
 const size_t N,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr X,
 const size_t incX,
 const typename Field::Element beta,
 typename Field::Element_ptr Y,
 const size_t incY,
 MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::LazyTag > & H) [inline]
```

**11.1.3.68 fgemv()** [5/19]

```
template<class Field >
Field::Element_ptr fgemv (
 const Field & F,
 const FFLAS_TRANSPOSE TransA,
 const size_t M,
 const size_t N,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr X,
 const size_t incX,
 const typename Field::Element beta,
 typename Field::Element_ptr Y,
 const size_t incY) [inline]
```

finite prime Field GEneral Matrix Vector multiplication.

Computes  $Y \leftarrow \alpha \text{op}(A)X + \beta Y$ .

**Parameters**

|     |          |                                                       |
|-----|----------|-------------------------------------------------------|
|     | $F$      | field                                                 |
|     | $TransA$ | if $TransA == FflasTrans$ then $\text{op}(A) = A^t$ . |
|     | $M$      | rows                                                  |
|     | $N$      | cols                                                  |
|     | $alpha$  | scalar                                                |
|     | $A$      | dense matrix of size $M \times N$                     |
|     | $lda$    | leading dimension of $A$                              |
|     | $X$      | dense vector of size $N$                              |
|     | $incX$   | stride of $X$                                         |
|     | $beta$   | scalar                                                |
| out | $Y$      | dense vector of size $M$                              |
|     | $incY$   | stride of $Y$                                         |

**11.1.3.69 fgemv()** [6/19]

```
Givaro::ZRing< int64_t >::Element_ptr fgemv (
 const Givaro::ZRing< int64_t > & F,
 const FFLAS_TRANSPOSE ta,
 const size_t M,
 const size_t N,
 const int64_t alpha,
 const int64_t * A,
 const size_t lda,
 const int64_t * X,
 const size_t incX,
 const int64_t beta,
 int64_t * Y,
 const size_t incY,
 MMHelper< Givaro::ZRing< int64_t >, MMHelperAlgo::Classic, ModeCategories::DefaultTag
> & H) [inline]
```

**11.1.3.70 fgemv()** [7/19]

```
Givaro::DoubleDomain::Element_ptr fgemv (
 const Givaro::DoubleDomain & F,
 const FFLAS_TRANSPOSE ta,
 const size_t M,
 const size_t N,
 const Givaro::DoubleDomain::Element alpha,
 const Givaro::DoubleDomain::ConstElement_ptr A,
 const size_t lda,
 const Givaro::DoubleDomain::ConstElement_ptr X,
 const size_t incX,
 const Givaro::DoubleDomain::Element beta,
 Givaro::DoubleDomain::Element_ptr Y,
 const size_t incY,
 MMHelper< Givaro::DoubleDomain, MMHelperAlgo::Classic, ModeCategories::DefaultTag
> & H) [inline]
```

**11.1.3.71 fgemv()** [8/19]

```
template<class Field >
Field::Element_ptr fgemv (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const size_t M,
 const size_t N,
 const typename Field::Element alpha,
 const typename Field::ConstElement_ptr A,
 const size_t lda,
 const typename Field::ConstElement_ptr X,
 const size_t incX,
 const typename Field::Element beta,
 typename Field::Element_ptr Y,
 const size_t incY,
 MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::DefaultBoundedTag > & H)
[inline]
```

**11.1.3.72 fgemv()** [9/19]

```
Givaro::FloatDomain::Element_ptr fgemv (
 const Givaro::FloatDomain & F,
 const FFLAS_TRANSPOSE ta,
 const size_t M,
 const size_t N,
 const Givaro::FloatDomain::Element alpha,
 const Givaro::FloatDomain::ConstElement_ptr A,
 const size_t lda,
 const Givaro::FloatDomain::ConstElement_ptr X,
 const size_t incX,
 const Givaro::FloatDomain::Element beta,
 Givaro::FloatDomain::Element_ptr Y,
 const size_t incY,
 MMHelper< Givaro::FloatDomain, MMHelperAlgo::Classic, ModeCategories::DefaultTag
> & H) [inline]
```

**11.1.3.73 fgemv()** [10/19]

```
template<class Field , class Cut , class Param >
Field::Element_ptr fgemv (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const size_t m,
 const size_t n,
 const typename Field::Element alpha,
 const typename Field::ConstElement_ptr A,
 const size_t lda,
 const typename Field::ConstElement_ptr X,
 const size_t incX,
 const typename Field::Element beta,
 typename Field::Element_ptr Y,
 const size_t incY,
 ParSeqHelper::Parallel< Cut, Param > & parH)
```

**11.1.3.74 fgemv()** [11/19]

```
template<class Field >
Field::Element_ptr fgemv (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const size_t m,
 const size_t n,
 const typename Field::Element alpha,
 const typename Field::ConstElement_ptr A,
 const size_t lda,
 const typename Field::ConstElement_ptr X,
 const size_t incX,
 const typename Field::Element beta,
 typename Field::Element_ptr Y,
 const size_t incY,
 ParSeqHelper::Sequential & seqH)
```

**11.1.3.75 fgemv()** [12/19]

```
FFPACK::rns_double::Element_ptr fgemv (
 const FFPACK::RNSInteger< FFPACK::rns_double > & F,
 const FFLAS_TRANSPOSE ta,
 const size_t M,
 const size_t N,
 const FFPACK::rns_double::Element alpha,
 FFPACK::rns_double::ConstElement_ptr A,
 const size_t lda,
 FFPACK::rns_double::ConstElement_ptr X,
 const size_t incX,
 const FFPACK::rns_double::Element beta,
 FFPACK::rns_double::Element_ptr Y,
 const size_t incY,
 MMHelper< FFPACK::RNSInteger< FFPACK::rns_double >, MMHelperAlgo::Classic, ModeCategories::Default > & H) [inline]
```

**11.1.3.76 fgemv()** [13/19]

```
FFPACK::rns_double::Element_ptr fgemv (
 const FFPACK::RNSIntegerMod< FFPACK::rns_double > & F,
 const FFLAS_TRANSPOSE ta,
 const size_t M,
 const size_t N,
 const FFPACK::rns_double::Element alpha,
 FFPACK::rns_double::ConstElement_ptr A,
 const size_t lda,
 FFPACK::rns_double::ConstElement_ptr X,
 const size_t incX,
 const FFPACK::rns_double::Element beta,
 FFPACK::rns_double::Element_ptr Y,
 const size_t incY,
 MMHelper< FFPACK::RNSIntegerMod< FFPACK::rns_double >, MMHelperAlgo::Classic,
 ModeCategories::DefaultTag > & H) [inline]
```

**11.1.3.77 fgemv()** [14/19]

```
Givaro::Integer * fgemv (
 const Givaro::ZRing< Givaro::Integer > & F,
 const FFLAS_TRANSPOSE ta,
 const size_t m,
 const size_t n,
 const Givaro::Integer alpha,
 Givaro::Integer * A,
 const size_t lda,
 Givaro::Integer * X,
 const size_t ldx,
 Givaro::Integer beta,
 Givaro::Integer * Y,
 const size_t ldy,
 MMHelper< Givaro::ZRing< Givaro::Integer >, MMHelperAlgo::Classic, ModeCategories::ConvertTo<
 ElementCategories::RNSElementTag > > & H) [inline]
```

**11.1.3.78 fgemv()** [15/19]

```
Givaro::Integer * fgemv (
 const Givaro::Modular< Givaro::Integer > & F,
 const FFLAS_TRANSPOSE ta,
 const size_t m,
 const size_t n,
 const Givaro::Integer alpha,
 Givaro::Integer * A,
 const size_t lda,
 Givaro::Integer * X,
 const size_t ldx,
 Givaro::Integer beta,
 Givaro::Integer * Y,
 const size_t ldy,
 MMHelper< Givaro::Modular< Givaro::Integer >, MMHelperAlgo::Classic, ModeCategories::ConvertTo<
 ElementCategories::RNSElementTag > > & H) [inline]
```

**11.1.3.79 fgemv()** [16/19]

```

template<size_t K1, size_t K2, class ParSeq >
RecInt::ruint< K1 > * fgemv (
 const Givaro::Modular< RecInt::ruint< K1 >, RecInt::ruint< K2 > > & F,
 const FFLAS_TRANSPOSE ta,
 const size_t m,
 const size_t n,
 const RecInt::ruint< K1 > alpha,
 const RecInt::ruint< K1 > * A,
 const size_t lda,
 const RecInt::ruint< K1 > * X,
 const size_t incx,
 RecInt::ruint< K1 > beta,
 RecInt::ruint< K1 > * Y,
 const size_t incy,
 MMHelper< Givaro::Modular< RecInt::ruint< K1 >, RecInt::ruint< K2 > >, MMHelperAlgo::Classic,
 ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeq > & H) [inline]

```

**11.1.3.80 fger()** [1/12]

```

template<class Field >
void fger (
 const Field & F,
 const size_t M,
 const size_t N,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr x,
 const size_t incx,
 typename Field::ConstElement_ptr y,
 const size_t incy,
 typename Field::Element_ptr A,
 const size_t lda) [inline]

```

fger: rank one update of a general matrix

Computes  $A \leftarrow \alpha x y^T + A$

**Parameters**

|         |          |                                                    |
|---------|----------|----------------------------------------------------|
|         | $F$      | field                                              |
|         | $M$      | rows                                               |
|         | $N$      | cols                                               |
|         | $\alpha$ | scalar                                             |
| in, out | $A$      | dense matrix of size MxN and leading dimension lda |
|         | $lda$    | leading dimension of A                             |
|         | $x$      | dense vector of size M                             |
|         | $incx$   | stride of X                                        |
|         | $y$      | dense vector of size N                             |
|         | $incy$   | stride of Y                                        |

**11.1.3.81 fger()** [2/12]

```

template<class Field >
void fger (
 const Field & F,
 const size_t M,
 const size_t N,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr x,
 const size_t incx,
 typename Field::ConstElement_ptr y,
 const size_t incy,
 typename Field::Element_ptr A,
 const size_t lda,
 MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::ConvertTo< ElementCategories::MachineFlo
> > & H) [inline]

```

**11.1.3.82 fger()** [3/12]

```

template<class Field , class AnyTag >
void fger (
 const Field & F,
 const size_t M,
 const size_t N,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr x,
 const size_t incx,
 typename Field::ConstElement_ptr y,
 const size_t incy,
 typename Field::Element_ptr A,
 const size_t lda,
 MMHelper< Field, MMHelperAlgo::Classic, AnyTag > & H) [inline]

```

**11.1.3.83 fger()** [4/12]

```

void fger (
 const Givaro::DoubleDomain & F,
 const size_t M,
 const size_t N,
 const Givaro::DoubleDomain::Element alpha,
 const Givaro::DoubleDomain::ConstElement_ptr x,
 const size_t incx,
 const Givaro::DoubleDomain::ConstElement_ptr y,
 const size_t incy,
 Givaro::DoubleDomain::Element_ptr A,
 const size_t lda,
 MMHelper< Givaro::DoubleDomain, MMHelperAlgo::Classic, ModeCategories::DefaultTag
> & H) [inline]

```

**11.1.3.84 fger()** [5/12]

```

template<class Field >
void fger (
 const Field & F,
 const size_t M,
 const size_t N,
 const typename Field::Element alpha,
 const typename Field::ConstElement_ptr x,
 const size_t incx,
 const typename Field::ConstElement_ptr y,
 const size_t incy,
 typename Field::Element_ptr A,
 const size_t lda,
 MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::DefaultBoundedTag > & H)
[inline]

```

**11.1.3.85 fger()** [6/12]

```

void fger (
 const Givaro::FloatDomain & F,
 const size_t M,
 const size_t N,
 const Givaro::FloatDomain::Element alpha,
 const Givaro::FloatDomain::ConstElement_ptr x,
 const size_t incx,
 const Givaro::FloatDomain::ConstElement_ptr y,
 const size_t incy,
 Givaro::FloatDomain::Element_ptr A,
 const size_t lda,
 MMHelper< Givaro::FloatDomain, MMHelperAlgo::Classic, ModeCategories::DefaultTag
> & H) [inline]

```

**11.1.3.86 fger()** [7/12]

```

template<class Field >
void fger (
 const Field & F,
 const size_t M,
 const size_t N,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr x,
 const size_t incx,
 typename Field::ConstElement_ptr y,
 const size_t incy,
 typename Field::Element_ptr A,
 const size_t lda,
 MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::LazyTag > & H) [inline]

```



**11.1.3.87 fger()** [8/12]

```

template<class Field >
void fger (
 const Field & F,
 const size_t M,
 const size_t N,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr x,
 const size_t incx,
 typename Field::ConstElement_ptr y,
 const size_t incy,
 typename Field::Element_ptr A,
 const size_t lda,
 MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::DelayedTag > & H) [inline]

```

**11.1.3.88 fger()** [9/12]

```

void fger (
 const Givaro::Modular< Givaro::Integer > & F,
 const size_t M,
 const size_t N,
 const typename Givaro::Integer alpha,
 typename Givaro::Integer * x,
 const size_t incx,
 typename Givaro::Integer * y,
 const size_t incy,
 typename Givaro::Integer * A,
 const size_t lda,
 MMHelper< Givaro::Modular< Givaro::Integer >, MMHelperAlgo::Classic, ModeCategories::ConvertTo<
ElementCategories::RNSElementTag > > & H) [inline]

```

**11.1.3.89 fger()** [10/12]

```

template<typename RNS >
void fger (
 const FFPACK::RNSInteger< RNS > & F,
 const size_t M,
 const size_t N,
 const typename FFPACK::RNSInteger< RNS >::Element alpha,
 typename FFPACK::RNSInteger< RNS >::Element_ptr x,
 const size_t incx,
 typename FFPACK::RNSInteger< RNS >::Element_ptr y,
 const size_t incy,
 typename FFPACK::RNSInteger< RNS >::Element_ptr A,
 const size_t lda,
 MMHelper< FFPACK::RNSInteger< RNS >, MMHelperAlgo::Classic, ModeCategories::DefaultTag
> & H) [inline]

```

**11.1.3.90 fger()** [11/12]

```

template<typename RNS >
void fger (
 const FFPACK::RNSIntegerMod< RNS > & F,
 const size_t M,
 const size_t N,
 const typename FFPACK::RNSIntegerMod< RNS >::Element alpha,
 typename FFPACK::RNSIntegerMod< RNS >::Element_ptr x,
 const size_t incx,
 typename FFPACK::RNSIntegerMod< RNS >::Element_ptr y,
 const size_t incy,
 typename FFPACK::RNSIntegerMod< RNS >::Element_ptr A,
 const size_t lda,
 MMHelper< FFPACK::RNSIntegerMod< RNS >, MMHelperAlgo::Classic > & H) [inline]

```

**11.1.3.91 freduce()** [1/11]

```

template<class Field >
void freduce (
 const Field & F,
 const size_t n,
 typename Field::ConstElement_ptr Y,
 const size_t incY,
 typename Field::Element_ptr X,
 const size_t incX)

```

$\text{freduce } x \leftarrow y \bmod F.$

**Parameters**

|        |                     |
|--------|---------------------|
| $F$    | field               |
| $n$    | size of the vectors |
| $Y$    | vector of Element   |
| $incY$ | stride of Y         |
| $X$    | vector in F         |
| $incX$ | stride of X         |

**Bug** use cblas\_(d)scal when possible

**11.1.3.92 freduce()** [2/11]

```

template<class Field >
void freduce (
 const Field & F,
 const size_t n,
 typename Field::Element_ptr X,
 const size_t incX)

```

$\text{freduce } x \leftarrow x \bmod F.$

## Parameters

|        |                     |
|--------|---------------------|
| $F$    | field               |
| $n$    | size of the vectors |
| $X$    | vector in $F$       |
| $incX$ | stride of $X$       |

**Bug** use `cblas_(d)scal` when possible

**11.1.3.93 `freduce_constoverride()`** [1/2]

```
template<class Field >
void freduce_constoverride (
 const Field & F,
 const size_t m,
 typename Field::ConstElement_ptr A,
 const size_t incX)
```

**11.1.3.94 `finit()`** [1/8]

```
template<class Field , class ConstOtherElement_ptr >
void finit (
 const Field & F,
 const size_t n,
 ConstOtherElement_ptr Y,
 const size_t incY,
 typename Field::Element_ptr X,
 const size_t incX)
```

**11.1.3.95 `finit()`** [2/8]

```
template<class Field >
void finit (
 const Field & F,
 const size_t n,
 typename Field::Element_ptr X,
 const size_t incX)
```

`finit` Initializes  $X$  in  $F$ .

## Parameters

|        |                     |
|--------|---------------------|
| $F$    | field               |
| $n$    | size of the vectors |
| $X$    | vector in $F$       |
| $incX$ | stride of $X$       |

**11.1.3.96 freduce()** [3/11]

```
template<class Field >
void freduce (
 const Field & F,
 const size_t m,
 const size_t n,
 typename Field::Element_ptr A,
 const size_t lda)
```

$\text{freduce } A \leftarrow A \bmod F.$

**Parameters**

|       |                |
|-------|----------------|
| $F$   | field          |
| $m$   | number of rows |
| $n$   | number of cols |
| $A$   | matrix in F    |
| $lda$ | stride of A    |

**11.1.3.97 freduce()** [4/11]

```
template<class Field >
void freduce (
 const Field & F,
 const FFLAS_UPLO UpLo,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda)
```

freduce for square symmetric matrices

**11.1.3.98 pfreduce()**

```
template<class Field >
void pfreduce (
 const Field & F,
 const size_t m,
 const size_t n,
 typename Field::Element_ptr A,
 const size_t lda,
 const size_t numths)
```

**11.1.3.99 freduce()** [5/11]

```
template<class Field >
void freduce (
 const Field & F,
 const size_t m,
 const size_t n,
 typename Field::ConstElement_ptr B,
```

```
const size_t ldb,
typename Field::Element_ptr A,
const size_t lda)
```

freduce  $A \leftarrow B \bmod F$ .

## Parameters

|       |                                |
|-------|--------------------------------|
| $F$   | field                          |
| $m$   | number of rows                 |
| $n$   | number of cols                 |
| $A$   | matrix in $F$                  |
| $lda$ | stride of $A$                  |
| $B$   | matrix in <code>Element</code> |
| $ldb$ | stride of $B$                  |

**11.1.3.100 `freduce_constoverride()` [2/2]**

```
template<class Field >
void freduce_constoverride (
 const Field & F,
 const size_t m,
 const size_t n,
 typename Field::ConstElement_ptr A,
 const size_t lda)
```

**11.1.3.101 `finit()` [3/8]**

```
template<class Field , class OtherElement_ptr >
void finit (
 const Field & F,
 const size_t m,
 const size_t n,
 const OtherElement_ptr B,
 const size_t ldb,
 typename Field::Element_ptr A,
 const size_t lda)
```

$\text{finit } A \leftarrow B \bmod F.$

## Parameters

|       |                                     |
|-------|-------------------------------------|
| $F$   | field                               |
| $m$   | number of rows                      |
| $n$   | number of cols                      |
| $A$   | matrix in $F$                       |
| $lda$ | stride of $A$                       |
| $B$   | matrix in <code>OtherElement</code> |
| $ldb$ | stride of $B$                       |

**11.1.3.102 `finit()` [4/8]**

```
template<class Field >
void finit (
 const Field & F,
```

```

const size_t m,
const size_t n,
typename Field::Element_ptr A,
const size_t lda)

```

#### 11.1.3.103 freduce() [6/11]

```

template<>
void freduce (
 const FFPACK::RNSIntegerMod< FFPACK::rns_double > & F,
 const size_t n,
 FFPACK::RNSIntegerMod< FFPACK::rns_double >::Element_ptr A,
 size_t inc) [inline]

```

#### 11.1.3.104 freduce() [7/11]

```

template<>
void freduce (
 const FFPACK::RNSIntegerMod< FFPACK::rns_double > & F,
 const size_t m,
 const size_t n,
 FFPACK::rns_double::Element_ptr A,
 size_t lda) [inline]

```

#### 11.1.3.105 freivalds()

```

template<class Field >
bool freivalds (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr B,
 const size_t ldb,
 typename Field::ConstElement_ptr C,
 const size_t ldc) [inline]

```

freivalds: **F**reivalds **G**eneral **M**atrix **M**ultiply **R**andom **C**heck.

Randomly Checks  $C = \alpha \text{op}(A) \times \text{op}(B)$

##### Parameters

|           |                                                                                      |
|-----------|--------------------------------------------------------------------------------------|
| <i>F</i>  | field.                                                                               |
| <i>ta</i> | if <code>ta==FflasTrans</code> then $\text{op}(A) = A^t$ , else $\text{op}(A) = A$ , |
| <i>tb</i> | same for matrix B                                                                    |
| <i>m</i>  | see A                                                                                |
| <i>n</i>  | see B                                                                                |

## Parameters

|          |                                |
|----------|--------------------------------|
| $k$      | see A                          |
| $\alpha$ | scalar                         |
| $A$      | $\text{op}(A)$ is $m \times k$ |
| $B$      | $\text{op}(B)$ is $k \times n$ |
| $C$      | $C$ is $m \times n$            |
| $lda$    | leading dimension of A         |
| $ldb$    | leading dimension of B         |
| $ldc$    | leading dimension of C         |

**11.1.3.106 fscaln()** [1/10]

```
template<class Field >
void fscaln (
 const Field & F,
 const size_t n,
 const typename Field::Element alpha,
 typename Field::Element_ptr X,
 const size_t incX) [inline]
```

$\text{fscaln } x \leftarrow \alpha \cdot x.$

## Parameters

|          |                     |
|----------|---------------------|
| $F$      | field               |
| $n$      | size of the vectors |
| $\alpha$ | scalar              |
| $X$      | vector in $F$       |
| $incX$   | stride of X         |

**Bug** use `cblas_(d)scal` when possible

**Todo** check if comparison with  $\pm 1, 0$  is necessary.

**11.1.3.107 fscal()** [1/10]

```
template<class Field >
void fscal (
 const Field & F,
 const size_t n,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr X,
 const size_t incX,
 typename Field::Element_ptr Y,
 const size_t incY) [inline]
```

$\text{fscal } y \leftarrow \alpha \cdot x.$



## Parameters

|     |          |                     |
|-----|----------|---------------------|
|     | $F$      | field               |
|     | $n$      | size of the vectors |
|     | $\alpha$ | scalar              |
| in  | $X$      | vector in $F$       |
|     | $incX$   | stride of $X$       |
| out | $Y$      | vector in $F$       |
|     | $incY$   | stride of $Y$       |

**Bug** use `cblas_(d)scal` when possible

**Todo** check if comparison with  $\pm 1, 0$  is necessary.

#### 11.1.3.108 fscal() [2/10]

```
template<>
void fscal (
 const Givaro::DoubleDomain & ,
 const size_t N,
 const Givaro::DoubleDomain::Element a,
 Givaro::DoubleDomain::ConstElement_ptr x,
 const size_t incx,
 Givaro::DoubleDomain::Element_ptr y,
 const size_t incy) [inline]
```

#### 11.1.3.109 fscal() [3/10]

```
template<>
void fscal (
 const Givaro::FloatDomain & ,
 const size_t N,
 const Givaro::FloatDomain::Element a,
 Givaro::FloatDomain::ConstElement_ptr x,
 const size_t incx,
 Givaro::FloatDomain::Element_ptr y,
 const size_t incy) [inline]
```

#### 11.1.3.110 fscaln() [2/10]

```
template<>
void fscaln (
 const Givaro::DoubleDomain & ,
 const size_t N,
 const Givaro::DoubleDomain::Element a,
 Givaro::DoubleDomain::Element_ptr y,
 const size_t incy) [inline]
```

**11.1.3.111 fscaln()** [3/10]

```
template<>
void fscaln (
 const Givaro::FloatDomain & ,
 const size_t N,
 const Givaro::FloatDomain::Element a,
 Givaro::FloatDomain::Element_ptr y,
 const size_t incy) [inline]
```

**11.1.3.112 fscaln()** [4/10]

```
template<class Field >
void fscaln (
 const Field & F,
 const size_t m,
 const size_t n,
 const typename Field::Element alpha,
 typename Field::Element_ptr A,
 const size_t lda) [inline]
```

$\text{fscaln } A \leftarrow a \cdot A.$

**Parameters**

|          |                  |
|----------|------------------|
| $F$      | field            |
| $m$      | number of rows   |
| $n$      | number of cols   |
| $\alpha$ | homotecie scalar |
| $A$      | matrix in F      |
| $lda$    | stride of A      |

**11.1.3.113 fscal()** [4/10]

```
template<class Field >
void fscal (
 const Field & F,
 const size_t m,
 const size_t n,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::Element_ptr B,
 const size_t ldb) [inline]
```

$\text{fscal } B \leftarrow a \cdot A.$

**Parameters**

|  |     |                |
|--|-----|----------------|
|  | $F$ | field          |
|  | $m$ | number of rows |
|  | $n$ | number of cols |

## Parameters

|     |              |                  |
|-----|--------------|------------------|
|     | <i>alpha</i> | homotecie scalar |
| in  | <i>A</i>     | matrix in F      |
|     | <i>lda</i>   | stride of A      |
| out | <i>B</i>     | matrix in F      |
|     | <i>ldb</i>   | stride of B      |

**11.1.3.114 fscaln()** [5/10]

```
template<>
void fscaln (
 const FFPACK::RNSInteger< FFPACK::rns_double > & F,
 const size_t n,
 const FFPACK::rns_double::Element alpha,
 FFPACK::rns_double::Element_ptr A,
 const size_t inc) [inline]
```

**11.1.3.115 fscal()** [5/10]

```
template<>
void fscal (
 const FFPACK::RNSInteger< FFPACK::rns_double > & F,
 const size_t n,
 const FFPACK::rns_double::Element alpha,
 FFPACK::rns_double::ConstElement_ptr A,
 const size_t Ainc,
 FFPACK::rns_double::Element_ptr B,
 const size_t Binc) [inline]
```

**11.1.3.116 fscaln()** [6/10]

```
template<>
void fscaln (
 const FFPACK::RNSInteger< FFPACK::rns_double > & F,
 const size_t m,
 const size_t n,
 const FFPACK::rns_double::Element alpha,
 FFPACK::rns_double::Element_ptr A,
 const size_t lda) [inline]
```

**11.1.3.117 fscal()** [6/10]

```
template<>
void fscal (
 const FFPACK::RNSInteger< FFPACK::rns_double > & F,
 const size_t m,
 const size_t n,
 const FFPACK::rns_double::Element alpha,
 FFPACK::rns_double::ConstElement_ptr A,
 const size_t lda,
 FFPACK::rns_double::Element_ptr B,
 const size_t ldb) [inline]
```

**11.1.3.118 fscaln()** [7/10]

```

template<>
void fscaln (
 const FFPACK::RNSIntegerMod< FFPACK::rns_double > & F,
 const size_t n,
 const typename FFPACK::RNSIntegerMod< FFPACK::rns_double >::Element alpha,
 typename FFPACK::RNSIntegerMod< FFPACK::rns_double >::Element_ptr A,
 const size_t inc) [inline]

```

**11.1.3.119 fscal()** [7/10]

```

template<>
void fscal (
 const FFPACK::RNSIntegerMod< FFPACK::rns_double > & F,
 const size_t n,
 const FFPACK::rns_double::Element alpha,
 FFPACK::rns_double::ConstElement_ptr A,
 const size_t Ainc,
 FFPACK::rns_double::Element_ptr B,
 const size_t Binc) [inline]

```

**11.1.3.120 fscaln()** [8/10]

```

template<>
void fscaln (
 const FFPACK::RNSIntegerMod< FFPACK::rns_double > & F,
 const size_t m,
 const size_t n,
 const FFPACK::rns_double::Element alpha,
 FFPACK::rns_double::Element_ptr A,
 const size_t lda) [inline]

```

**11.1.3.121 fscal()** [8/10]

```

template<>
void fscal (
 const FFPACK::RNSIntegerMod< FFPACK::rns_double > & F,
 const size_t m,
 const size_t n,
 const FFPACK::rns_double::Element alpha,
 FFPACK::rns_double::ConstElement_ptr A,
 const size_t lda,
 FFPACK::rns_double::Element_ptr B,
 const size_t ldb) [inline]

```

**11.1.3.122 fsyr2k()**

```
template<class Field >
Field::Element_ptr fsyr2k (
 const Field & F,
 const FFLAS_UPLO UpLo,
 const FFLAS_TRANSPOSE trans,
 const size_t n,
 const size_t k,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr B,
 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc) [inline]
```

fsyr2k: Symmetric Rank 2K update

Computes the Lower or Upper triangular part of  $C = \alpha(A \times B^T + B \times A^T) + \beta C$  or  $C = \alpha(A^T \times B + B^T \times A) + \beta C$

**Parameters**

|              |                                                                                                                                                              |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>F</i>     | field.                                                                                                                                                       |
| <i>UpLo</i>  | whether to compute the upper or the lower triangular part of the symmetric matrix C                                                                          |
| <i>trans</i> | if <code>ta==FflasNoTrans</code> then compute $C = \alpha(A \times B^T + B \times A^T) + \beta C$ , else $C = \alpha(A^T \times B + B^T \times A) + \beta C$ |
| <i>n</i>     | order of matrix C                                                                                                                                            |
| <i>k</i>     | see A                                                                                                                                                        |
| <i>alpha</i> | scalar                                                                                                                                                       |
| <i>A</i>     | A is $n \times k$ (FflasNoTrans) or A is $k \times n$ (FflasTrans)                                                                                           |
| <i>lda</i>   | leading dimension of A                                                                                                                                       |
| <i>beta</i>  | scalar                                                                                                                                                       |
| <i>C</i>     | C is $n \times n$                                                                                                                                            |
| <i>ldc</i>   | leading dimension of C                                                                                                                                       |

**Warning**

$\alpha$  must be invertible

**11.1.3.123 fsyrk()** [1/16]

```
template<class Field >
Field::Element_ptr fsyrk (
 const Field & F,
 const FFLAS_UPLO UpLo,
 const FFLAS_TRANSPOSE trans,
 const size_t n,
 const size_t k,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
```

```

const size_t lda,
const typename Field::Element beta,
typename Field::Element_ptr C,
const size_t ldc) [inline]

```

fsyrk: Symmetric Rank K update

Computes the Lower or Upper triangular part of  $C = \alpha A \times A^T + \beta C$  or  $C = \alpha A^T \times A + \beta C$

#### Parameters

|              |                                                                                                                              |
|--------------|------------------------------------------------------------------------------------------------------------------------------|
| <i>F</i>     | field.                                                                                                                       |
| <i>UpLo</i>  | whether to compute the upper or the lower triangular part of the symmetric matrix C                                          |
| <i>trans</i> | if <code>ta==FflasNoTrans</code> then compute $C = \alpha A \times A^T + \beta C$ , else $C = \alpha A^T \times A + \beta C$ |
| <i>n</i>     | order of matrix C                                                                                                            |
| <i>k</i>     | see A                                                                                                                        |
| <i>alpha</i> | scalar                                                                                                                       |
| <i>A</i>     | <i>A</i> is $n \times k$ or <i>A</i> is $k \times n$                                                                         |
| <i>lda</i>   | leading dimension of A                                                                                                       |
| <i>beta</i>  | scalar                                                                                                                       |
| <i>C</i>     | <i>C</i> is $n \times n$                                                                                                     |
| <i>ldc</i>   | leading dimension of C                                                                                                       |

#### Warning

$\alpha$  must be invertible

#### 11.1.3.124 fsyrk() [2/16]

```

template<class Field >
Field::Element_ptr fsyrk (
 const Field & F,
 const FFLAS_UPLO UpLo,
 const FFLAS_TRANSPOSE trans,
 const size_t N,
 const size_t K,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 const ParSeqHelper::Sequential seq) [inline]

```

#### 11.1.3.125 fsyrk() [3/16]

```

template<class Field >
Field::Element_ptr fsyrk (
 const Field & F,
 const FFLAS_UPLO UpLo,
 const FFLAS_TRANSPOSE trans,

```

```

 const size_t N,
 const size_t K,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::DefaultTag > & H) [inline]

```

### 11.1.3.126 fsyrk() [4/16]

```

template<class Field >
Field::Element_ptr fsyrk (
 const Field & F,
 const FFLAS_UPLO UpLo,
 const FFLAS_TRANSPOSE trans,
 const size_t N,
 const size_t K,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::ConvertTo< ElementCategories::MachineFlo
>, ParSeqHelper::Sequential > & H) [inline]

```

### 11.1.3.127 fsyrk() [5/16]

```

template<class Field >
Field::Element_ptr fsyrk (
 const Field & F,
 const FFLAS_UPLO UpLo,
 const FFLAS_TRANSPOSE trans,
 const size_t N,
 const size_t K,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::DelayedTag > & H) [inline]

```

### 11.1.3.128 fsyrk() [6/16]

```

template<class Field >
Field::Element_ptr fsyrk (
 const Field & F,
 const FFLAS_UPLO UpLo,
 const FFLAS_TRANSPOSE trans,
 const size_t N,

```

```

 const size_t K,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::LazyTag > & H) [inline]

```

### 11.1.3.129 fsyrk() [7/16]

```

template<class Field , typename Mode >
Field::Element_ptr fsyrk (
 const Field & F,
 const FFLAS_UPLO UpLo,
 const FFLAS_TRANSPOSE trans,
 const size_t N,
 const size_t K,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 MMHelper< Field, MMHelperAlgo::DivideAndConquer, Mode > & H) [inline]

```

### 11.1.3.130 fsyrk() [8/16]

```

template<class Field >
Field::Element_ptr fsyrk (
 const Field & F,
 const FFLAS_UPLO UpLo,
 const FFLAS_TRANSPOSE trans,
 const size_t N,
 const size_t K,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 MMHelper< Field, MMHelperAlgo::Classic, ModeCategories::DefaultBoundedTag > & H)
[inline]

```

### 11.1.3.131 fsyrk() [9/16]

```

Givaro::FloatDomain::Element_ptr fsyrk (
 const Givaro::FloatDomain & F,
 const FFLAS_UPLO UpLo,
 const FFLAS_TRANSPOSE trans,
 const size_t N,
 const size_t K,
 const Givaro::FloatDomain::Element alpha,

```



```

 Givaro::FloatDomain::ConstElement_ptr A,
 const size_t lda,
 const Givaro::FloatDomain::Element beta,
 Givaro::FloatDomain::Element_ptr C,
 const size_t ldc,
 MMHelper< Givaro::FloatDomain, MMHelperAlgo::Classic, ModeCategories::DefaultTag
> & H) [inline]

```

### 11.1.3.132 fsyrk() [10/16]

```

Givaro::DoubleDomain::Element_ptr fsyrk (
 const Givaro::DoubleDomain & F,
 const FFLAS_UPLO UpLo,
 const FFLAS_TRANSPOSE trans,
 const size_t N,
 const size_t K,
 const Givaro::DoubleDomain::Element alpha,
 Givaro::DoubleDomain::ConstElement_ptr A,
 const size_t lda,
 const Givaro::DoubleDomain::Element beta,
 Givaro::DoubleDomain::Element_ptr C,
 const size_t ldc,
 MMHelper< Givaro::DoubleDomain, MMHelperAlgo::Classic, ModeCategories::DefaultTag
> & H) [inline]

```

### 11.1.3.133 fsyrk() [11/16]

```

template<class Field >
Field::Element_ptr fsyrk (
 const Field & F,
 const FFLAS_UPLO UpLo,
 const FFLAS_TRANSPOSE trans,
 const size_t n,
 const size_t k,
 const typename Field::Element alpha,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr D,
 const size_t incD,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 const size_t threshold = __FFLASFFPACK_FSYRK_THRESHOLD) [inline]

```

fsyrk: Symmetric Rank K update with diagonal scaling

Computes the Lower or Upper triangular part of  $C = \alpha A \times D \times A^T + \beta C$  or  $C = \alpha A^T \times D \times A + \beta C$  where  $D$  is a diagonal matrix. Matrix A is updated into  $D \times A$  (if trans = FflasTrans) or  $A \times D$  (if trans = FflasNoTrans).

#### Parameters

|              |                                                                                                                 |
|--------------|-----------------------------------------------------------------------------------------------------------------|
| <i>F</i>     | field.                                                                                                          |
| <i>UpLo</i>  | whether to compute the upper or the lower triangular part of the symmetric matrix C                             |
| <i>trans</i> | if ta==FflasNoTrans then compute $C = \alpha A \times A^T + \beta C$ , else $C = \alpha A^T \times A + \beta C$ |

## Parameters

|          |                                                                             |
|----------|-----------------------------------------------------------------------------|
| $n$      | order of matrix $C$                                                         |
| $k$      | see $A$                                                                     |
| $\alpha$ | scalar                                                                      |
| $A$      | $A$ is $n \times k$ or $A$ is $k \times n$                                  |
| $lda$    | leading dimension of $A$                                                    |
| $D$      | $D$ is $k \times k$ diagonal matrix, stored as a vector of $k$ coefficients |
| $lda$    | leading dimension of $A$                                                    |
| $\beta$  | scalar                                                                      |
| $C$      | $C$ is $n \times n$                                                         |
| $ldc$    | leading dimension of $C$                                                    |

## Warning

$\alpha$  must be invertible

## 11.1.3.134 fsyrk() [12/16]

```
template<class Field >
Field::Element_ptr fsyrk (
 const Field & F,
 const FFLAS_UPLO UpLo,
 const FFLAS_TRANSPOSE trans,
 const size_t N,
 const size_t K,
 const typename Field::Element alpha,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr D,
 const size_t incD,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 const ParSeqHelper::Sequential seq,
 const size_t threshold) [inline]
```

## 11.1.3.135 fsyrk() [13/16]

```
template<class Field , class Cut , class Param >
Field::Element_ptr fsyrk (
 const Field & F,
 const FFLAS_UPLO UpLo,
 const FFLAS_TRANSPOSE trans,
 const size_t N,
 const size_t K,
 const typename Field::Element alpha,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr D,
 const size_t incD,
```

```

const typename Field::Element beta,
typename Field::Element_ptr C,
const size_t ldc,
const ParSeqHelper::Parallel< Cut, Param > par,
const size_t threshold) [inline]

```

### 11.1.3.136 fsyrk() [14/16]

```

template<class Field >
Field::Element_ptr fsyrk (
 const Field & F,
 const FFLAS_UPLO UpLo,
 const FFLAS_TRANSPOSE trans,
 const size_t n,
 const size_t k,
 const typename Field::Element alpha,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr D,
 const size_t incD,
 const std::vector< bool > & twoBlock,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 const size_t threshold = __FFLASFFPACK_FSYRK_THRESHOLD) [inline]

```

fsyrk: Symmetric Rank K update with diagonal scaling

Computes the Lower or Upper triangular part of  $C = \alpha A \times \text{Delta} D \times A^T + \beta C$  or  $C = \alpha A^T \times \text{Delta} D \times A + \beta C$  where D is a diagonal matrix and Delta is a block diagonal with either 1 on the diagonal or 2x2 swap blocks Matrix A is updated into  $D \times A$  (if trans = FflasTrans) or  $A \times D$  (if trans = FflasNoTrans).

#### Parameters

|                  |                                                                                                                                               |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <i>F</i>         | field.                                                                                                                                        |
| <i>UpLo</i>      | whether to compute the upper or the lower triangular part of the symmetric matrix C                                                           |
| <i>trans</i>     | if ta==FflasNoTrans then compute $C = \alpha A \text{Delta} D \times A^T + \beta C$ , else $C = \alpha A^T \text{Delta} D \times A + \beta C$ |
| <i>n</i>         | see B                                                                                                                                         |
| <i>k</i>         | see A                                                                                                                                         |
| <i>alpha</i>     | scalar                                                                                                                                        |
| <i>A</i>         | $A$ is $n \times k$ or $A$ is $k \times n$                                                                                                    |
| <i>lda</i>       | leading dimension of A                                                                                                                        |
| <i>D</i>         | $D$ is $k \times k$ diagonal matrix, stored as a vector of k coefficients                                                                     |
| <i>twoBlocks</i> | a vector boolean indicating the beginning of each 2x2 blocs in Delta                                                                          |
| <i>lda</i>       | leading dimension of A                                                                                                                        |
| <i>beta</i>      | scalar                                                                                                                                        |
| <i>C</i>         | $C$ is $n \times n$                                                                                                                           |
| <i>ldc</i>       | leading dimension of C                                                                                                                        |

#### Warning

$\alpha$  must be invertible

**11.1.3.137 computeS1S2()**

```

template<class Field , class FieldTrait >
void computeS1S2 (
 const Field & F,
 const FFLAS_TRANSPOSE trans,
 const size_t N,
 const size_t K,
 const typename Field::Element x,
 const typename Field::Element y,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr S,
 const size_t lds,
 typename Field::Element_ptr T,
 const size_t ldt,
 MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > & WH) [inline]

```

**11.1.3.138 fsyrk() [15/16]**

```

template<class Field >
Field::Element_ptr fsyrk (
 const Field & F,
 const FFLAS_UPLO UpLo,
 const FFLAS_TRANSPOSE trans,
 const size_t N,
 const size_t K,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 MMHelper< Field, MMHelperAlgo::Winograd, ModeCategories::DelayedTag, ParSeqHelper::Sequential
> & H) [inline]

```

**11.1.3.139 fsyrk() [16/16]**

```

template<class Field , class Mode >
Field::Element_ptr fsyrk (
 const Field & F,
 const FFLAS_UPLO UpLo,
 const FFLAS_TRANSPOSE trans,
 const size_t N,
 const size_t K,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 MMHelper< Field, MMHelperAlgo::Winograd, Mode > & H) [inline]

```

**11.1.3.140 fsyrk\_strassen()** [1/2]

```
template<class Field , class FieldTrait >
Field::Element_ptr fsyrk_strassen (
 const Field & F,
 const FFLAS_UPLO uplo,
 const FFLAS_TRANSPOSE trans,
 const size_t N,
 const size_t K,
 const typename Field::Element y1,
 const typename Field::Element y2,
 const typename Field::Element alpha,
 typename Field::Element_ptr A,
 const size_t lda,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > & WH) [inline]
```

**11.1.3.141 ftrmm()** [1/3]

```
template<class Field >
void ftrmm (
 const Field & F,
 const FFLAS_SIDE Side,
 const FFLAS_UPLO Uplo,
 const FFLAS_TRANSPOSE TransA,
 const FFLAS_DIAG Diag,
 const size_t M,
 const size_t N,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::Element_ptr B,
 const size_t ldb) [inline]
```

ftrmm: **T**Riangular **M**atrix **M**ultiply.

Computes  $B \leftarrow \alpha \text{op}(A)B$  or  $B \leftarrow \alpha B \text{op}(A)$ .

**Parameters**

|               |                                                                                            |
|---------------|--------------------------------------------------------------------------------------------|
| <i>F</i>      | field                                                                                      |
| <i>Side</i>   | if Side==FflasLeft then $B \leftarrow \alpha \text{op}(A)B$ is computed.                   |
| <i>Uplo</i>   | if Uplo==FflasUpper then A is upper triangular                                             |
| <i>TransA</i> | if TransA==FflasTrans then $\text{op}(A) = A^t$ .                                          |
| <i>Diag</i>   | if Diag==FflasUnit then A is implicitly unit.                                              |
| <i>M</i>      | rows of B                                                                                  |
| <i>N</i>      | cols of B                                                                                  |
| <i>alpha</i>  | scalar                                                                                     |
| <i>A</i>      | triangular matrix. If Side==FflasLeft then A is $N \times N$ , otherwise A is $M \times M$ |
| <i>lda</i>    | leading dim of A                                                                           |
| <i>B</i>      | matrix of size MxN                                                                         |
| <i>ldb</i>    | leading dim of B                                                                           |

**11.1.3.142 ftrmm()** [2/3]

```

template<class Field >
void ftrmm (
 const Field & F,
 const FFLAS_SIDE Side,
 const FFLAS_UPLO Uplo,
 const FFLAS_TRANSPOSE TransA,
 const FFLAS_DIAG Diag,
 const size_t M,
 const size_t N,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr B,
 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc) [inline]

```

ftrmm: **TR**iangular **M**atrix **M**ultiply with 3 operands Computes  $C \leftarrow \alpha \text{op}(A)B + \beta C$  or  $C \leftarrow \alpha B \text{op}(A) + \beta C$ .

**Parameters**

|               |                                                                                            |
|---------------|--------------------------------------------------------------------------------------------|
| <i>F</i>      | field                                                                                      |
| <i>Side</i>   | if Side==FflasLeft then $B \leftarrow \alpha \text{op}(A)B$ is computed.                   |
| <i>Uplo</i>   | if Uplo==FflasUpper then A is upper triangular                                             |
| <i>TransA</i> | if TransA==FflasTrans then $\text{op}(A) = A^t$ .                                          |
| <i>Diag</i>   | if Diag==FflasUnit then A is implicitly unit.                                              |
| <i>M</i>      | rows of B                                                                                  |
| <i>N</i>      | cols of B                                                                                  |
| <i>alpha</i>  | scalar                                                                                     |
| <i>A</i>      | triangular matrix. If Side==FflasLeft then A is $N \times N$ , otherwise A is $M \times M$ |
| <i>lda</i>    | leading dim of A                                                                           |
| <i>B</i>      | matrix of size MxN                                                                         |
| <i>ldb</i>    | leading dim of B                                                                           |
| <i>beta</i>   | scalar                                                                                     |
| <i>C</i>      | matrix of size MxN                                                                         |
| <i>ldc</i>    | leading dim of C                                                                           |

**11.1.3.143 ftrsm()** [1/9]

```

template<class Field >
void ftrsm (
 const Field & F,
 const FFLAS_SIDE Side,
 const FFLAS_UPLO Uplo,
 const FFLAS_TRANSPOSE TransA,
 const FFLAS_DIAG Diag,
 const size_t M,
 const size_t N,
 const typename Field::Element alpha,

```

```

 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr B,
 const size_t ldb) [inline]

```

#### 11.1.3.144 ftrsm() [2/9]

```

template<class Field >
void ftrsm (
 const Field & F,
 const FFLAS_SIDE Side,
 const FFLAS_UPLO Uplo,
 const FFLAS_TRANSPOSE TransA,
 const FFLAS_DIAG Diag,
 const size_t M,
 const size_t N,
 const typename Field::Element alpha,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr B,
 const size_t ldb,
 const ParSeqHelper::Sequential & PSH) [inline]

```

#### 11.1.3.145 ftrsm() [3/9]

```

template<class Field , class Cut , class Param >
void ftrsm (
 const Field & F,
 const FFLAS_SIDE Side,
 const FFLAS_UPLO Uplo,
 const FFLAS_TRANSPOSE TransA,
 const FFLAS_DIAG Diag,
 const size_t M,
 const size_t N,
 const typename Field::Element alpha,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr B,
 const size_t ldb,
 const ParSeqHelper::Parallel< Cut, Param > & PSH) [inline]

```

#### 11.1.3.146 ftrsm() [4/9]

```

template<class Field , class ParSeqTrait = ParSeqHelper::Sequential>
void ftrsm (
 const Field & F,
 const FFLAS_SIDE Side,
 const FFLAS_UPLO Uplo,
 const FFLAS_TRANSPOSE TransA,
 const FFLAS_DIAG Diag,
 const size_t M,
 const size_t N,
 const typename Field::Element alpha,

```

```

typename Field::Element_ptr A,
const size_t lda,
typename Field::Element_ptr B,
const size_t ldb,
TRSMHelper< StructureHelper::Recursive, ParSeqTrait > & H) [inline]

```

#### 11.1.3.147 ftrsm() [5/9]

```

void ftrsm (
 const Givaro::Modular< Givaro::Integer > & F,
 const FFLAS_SIDE Side,
 const FFLAS_UPLO Uplo,
 const FFLAS_TRANSPOSE TransA,
 const FFLAS_DIAG Diag,
 const size_t M,
 const size_t N,
 const Givaro::Integer alpha,
 const Givaro::Integer * A,
 const size_t lda,
 Givaro::Integer * B,
 const size_t ldb) [inline]

```

#### 11.1.3.148 cblas\_impstrsm()

```

void cblas_impstrsm (
 const enum FFLAS_ORDER Order,
 const enum FFLAS_SIDE Side,
 const enum FFLAS_UPLO Uplo,
 const enum FFLAS_TRANSPOSE TransA,
 const enum FFLAS_DIAG Diag,
 const int M,
 const int N,
 const FFPACK::rns_double_elt alpha,
 FFPACK::rns_double_elt_cstptr A,
 const int lda,
 FFPACK::rns_double_elt_ptr B,
 const int ldb) [inline]

```

#### 11.1.3.149 ftrsv() [1/2]

```

template<class Field >
void ftrsv (
 const Field & F,
 const FFLAS_UPLO Uplo,
 const FFLAS_TRANSPOSE TransA,
 const FFLAS_DIAG Diag,
 const size_t N,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::Element_ptr X,
 int incX) [inline]

```

ftrsv: TRIangular System solve with Vector Computes  $X \leftarrow \text{op}(A^{-1})X$



## Parameters

|               |                                                    |
|---------------|----------------------------------------------------|
| <i>F</i>      | field                                              |
| <i>X</i>      | vector of size N on a field F                      |
| <i>incX</i>   | stride of X                                        |
| <i>A</i>      | a matrix of leading dimension lda and size N       |
| <i>lda</i>    | leading dimension of A                             |
| <i>N</i>      | number of rows or columns of A according to TransA |
| <i>TransA</i> | if TransA==FflasTrans then $\text{op}(A) = A^t$ .  |
| <i>Diag</i>   | if Diag==FflasUnit then A is unit.                 |
| <i>Uplo</i>   | if Uplo==FflasUpper then A is upper triangular     |

## 11.1.3.150 igemm\_()

```

void igemm_ (
 const enum FFLAS_ORDER Order,
 const enum FFLAS_TRANSPOSE TransA,
 const enum FFLAS_TRANSPOSE TransB,
 const size_t M,
 const size_t N,
 const size_t K,
 const int64_t alpha,
 const int64_t * A,
 const size_t lda,
 const int64_t * B,
 const size_t ldb,
 const int64_t beta,
 int64_t * C,
 const size_t ldc) [inline]

```

## 11.1.3.151 finit() [5/8]

```

template<class Field , class OtherElement_ptr >
void finit (
 const Field & F,
 const size_t n,
 const OtherElement_ptr Y,
 const size_t incY,
 typename Field::Element_ptr X,
 const size_t incX)

```

$\text{finit } x \leftarrow y \bmod F.$

## Parameters

|             |                        |
|-------------|------------------------|
| <i>F</i>    | field                  |
| <i>n</i>    | size of the vectors    |
| <i>Y</i>    | vector of OtherElement |
| <i>incY</i> | stride of Y            |
| <i>X</i>    | vector in F            |
| <i>incX</i> | stride of X            |

**Bug** use `cblas_(d)scal` when possible

#### 11.1.3.152 `fconvert()` [1/3]

```
template<class Field , class OtherElement_ptr >
void fconvert (
 const Field & F,
 const size_t n,
 OtherElement_ptr X,
 const size_t incX,
 typename Field::ConstElement_ptr Y,
 const size_t incY)
```

`fconvert`  $x \leftarrow y \bmod F$ .

##### Parameters

|        |                                     |
|--------|-------------------------------------|
| $F$    | field                               |
| $n$    | size of the vectors                 |
| $Y$    | vector of $F$                       |
| $incY$ | stride of $Y$                       |
| $X$    | vector in <code>OtherElement</code> |
| $incX$ | stride of $X$                       |

**Bug** use `cblas_(d)scal` when possible

#### 11.1.3.153 `fnegin()` [1/4]

```
template<class Field >
void fnegin (
 const Field & F,
 const size_t n,
 typename Field::Element_ptr X,
 const size_t incX)
```

`fnegin`  $x \leftarrow -x$ .

##### Parameters

|        |                     |
|--------|---------------------|
| $F$    | field               |
| $n$    | size of the vectors |
| $X$    | vector in $F$       |
| $incX$ | stride of $X$       |

**Bug** use `cblas_(d)scal` when possible

**11.1.3.154 fneg()** [1/4]

```
template<class Field >
void fneg (
 const Field & F,
 const size_t n,
 typename Field::ConstElement_ptr Y,
 const size_t incY,
 typename Field::Element_ptr X,
 const size_t incX)
```

$\text{fneg } x \leftarrow -y.$

**Parameters**

|        |                     |
|--------|---------------------|
| $F$    | field               |
| $n$    | size of the vectors |
| $X$    | vector in $F$       |
| $incX$ | stride of $X$       |
| $Y$    | vector in $F$       |
| $incY$ | stride of $Y$       |

**Bug** use `cblas_(d)scal` when possible

**11.1.3.155 fzero()** [1/5]

```
template<class Field >
void fzero (
 const Field & F,
 const size_t n,
 typename Field::Element_ptr X,
 const size_t incX)
```

$\text{fzero} : A \leftarrow 0.$

**Parameters**

|        |                            |
|--------|----------------------------|
| $F$    | field                      |
| $n$    | number of elements to zero |
| $X$    | vector in $F$              |
| $incX$ | stride of $X$              |

**11.1.3.156 frand()** [1/2]

```
template<class Field , class RandIter >
void frand (
 const Field & F,
 RandIter & G,
 const size_t n,
```

```

 typename Field::Element_ptr X,
 const size_t incX)

```

frand :  $A \leftarrow \text{random}$ .

#### Parameters

|        |                                 |
|--------|---------------------------------|
| $F$    | field                           |
| $G$    | randomiterator                  |
| $n$    | number of elements to randomize |
| $X$    | vector in $F$                   |
| $incX$ | stride of $X$                   |

#### 11.1.3.157 fiszero() [1/4]

```

template<class Field >
bool fiszero (
 const Field & F,
 const size_t n,
 typename Field::ConstElement_ptr X,
 const size_t incX)

```

fiszero : test  $X = 0$ .

#### Parameters

|        |                  |
|--------|------------------|
| $F$    | field            |
| $n$    | vector dimension |
| $X$    | vector in $F$    |
| $incX$ | increment of $X$ |

#### 11.1.3.158 fequal() [1/4]

```

template<class Field >
bool fequal (
 const Field & F,
 const size_t n,
 typename Field::ConstElement_ptr X,
 const size_t incX,
 typename Field::ConstElement_ptr Y,
 const size_t incY)

```

fequal : test  $X = Y$ .

#### Parameters

|        |                  |
|--------|------------------|
| $F$    | field            |
| $n$    | vector dimension |
| $X$    | vector in $F$    |
| $incX$ | increment of $X$ |
| $Y$    | vector in $F$    |
| $incY$ | increment of $Y$ |

**11.1.3.159 faxpby()** [1/2]

```
template<class Field >
void faxpby (
 const Field & F,
 const size_t N,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr X,
 const size_t incX,
 const typename Field::Element beta,
 typename Field::Element_ptr Y,
 const size_t incY)
```

$\text{faxpby} : y \leftarrow \alpha \cdot x + \beta \cdot y.$

**Parameters**

|         |               |                     |
|---------|---------------|---------------------|
|         | $F$           | field               |
|         | $N$           | size of the vectors |
|         | $\alpha$      | scalar              |
| in      | $X$           | vector in F         |
|         | $\text{incX}$ | stride of X         |
|         | $\beta$       | scalar              |
| in, out | $Y$           | vector in F         |
|         | $\text{incY}$ | stride of Y         |

**Note**

this is a catlas function

**11.1.3.160 fdot()** [9/11]

```
template<typename Field , class Cut , class Param >
Field::Element fdot (
 const Field & F,
 const size_t N,
 typename Field::ConstElement_ptr X,
 const size_t incX,
 typename Field::ConstElement_ptr Y,
 const size_t incY,
 const ParSeqHelper::Parallel< Cut, Param > par) [inline]
```

**11.1.3.161 fswap()** [1/2]

```
template<class Field >
void fswap (
 const Field & F,
 const size_t N,
 typename Field::Element_ptr X,
 const size_t incX,
 typename Field::Element_ptr Y,
 const size_t incY)
```

$\text{fswap} : X \leftrightarrow Y.$

**Bug** use cblas\_dswap when double

## Parameters

|        |                     |
|--------|---------------------|
| $F$    | field               |
| $N$    | size of the vectors |
| $X$    | vector in $F$       |
| $incX$ | stride of $X$       |
| $Y$    | vector in $F$       |
| $incY$ | stride of $Y$       |

**11.1.3.162 fzero()** [2/5]

```
template<class Field >
void fzero (
 const Field & F,
 const size_t m,
 const size_t n,
 typename Field::Element_ptr A,
 const size_t lda)
```

$fzero : A \leftarrow 0$ .

## Parameters

|       |                        |
|-------|------------------------|
| $F$   | field                  |
| $m$   | number of rows to zero |
| $n$   | number of cols to zero |
| $A$   | matrix in $F$          |
| $lda$ | stride of $A$          |

## Warning

may be buggy if Element is larger than int

**11.1.3.163 fzero()** [3/5]

```
template<class Field >
void fzero (
 const Field & F,
 const FFLAS_UPLO shape,
 const FFLAS_DIAG diag,
 const size_t n,
 typename Field::Element_ptr A,
 const size_t lda)
```

$fzero : A \leftarrow 0$  for a triangular matrix.

## Parameters

|         |                                |
|---------|--------------------------------|
| $F$     | field                          |
| $shape$ | shape of the triangular matrix |
| $m$     | number of rows to zero         |
| $n$     | number of cols to zero         |
| $A$     | matrix in $F$                  |
| $lda$   | stride of $A$                  |

**Warning**

may be buggy if Element is larger than int

**11.1.3.164 frand() [2/2]**

```
template<class Field , class RandIter >
void frand (
 const Field & F,
 RandIter & G,
 const size_t m,
 const size_t n,
 typename Field::Element_ptr A,
 const size_t lda)
```

frand :  $A \leftarrow \text{random}.$

**Parameters**

|       |                             |
|-------|-----------------------------|
| $F$   | field                       |
| $G$   | randomiterator              |
| $m$   | number of rows to randomize |
| $n$   | number of cols to randomize |
| $A$   | matrix in F                 |
| $lda$ | stride of A                 |

**11.1.3.165 fequal() [2/4]**

```
template<class Field >
bool fequal (
 const Field & F,
 const size_t m,
 const size_t n,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr B,
 const size_t ldb)
```

fequal : test  $A = B.$

**Parameters**

|       |                        |
|-------|------------------------|
| $F$   | field                  |
| $m$   | row dimension          |
| $n$   | column dimension       |
| $A$   | m x n matrix in F      |
| $lda$ | leading dimension of A |
| $B$   | m x n matrix in F      |
| $ldb$ | leading dimension of B |



**11.1.3.166 fiszero()** [2/4]

```
template<class Field >
bool fiszero (
 const Field & F,
 const size_t m,
 const size_t n,
 typename Field::ConstElement_ptr A,
 const size_t lda)
```

**fiszero** : test  $A = 0$ .

**Parameters**

|       |                          |
|-------|--------------------------|
| $F$   | field                    |
| $m$   | row dimension            |
| $n$   | column dimension         |
| $A$   | m x n matrix in $F$      |
| $lda$ | leading dimension of $A$ |

**11.1.3.167 fidentity()** [1/4]

```
template<class Field >
void fidentity (
 const Field & F,
 const size_t m,
 const size_t n,
 typename Field::Element_ptr A,
 const size_t lda,
 const typename Field::Element & d)
```

creates a diagonal matrix

**11.1.3.168 fidentity()** [2/4]

```
template<class Field >
void fidentity (
 const Field & F,
 const size_t m,
 const size_t n,
 typename Field::Element_ptr A,
 const size_t lda)
```

creates a diagonal matrix

**11.1.3.169 finit()** [6/8]

```
template<class Field , class OtherElement_ptr >
void finit (
 const Field & F,
 const size_t m,
 const size_t n,
 typename Field::Element_ptr A,
 const size_t lda)
```

**finit** Initializes  $A$  in  $F$ .

## Parameters

|       |                |
|-------|----------------|
| $F$   | field          |
| $m$   | number of rows |
| $n$   | number of cols |
| $A$   | matrix in $F$  |
| $lda$ | stride of $A$  |

**11.1.3.170 fconvert()** [2/3]

```
template<class Field , class OtherElement_ptr >
void fconvert (
 const Field & F,
 const size_t m,
 const size_t n,
 OtherElement_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr B,
 const size_t ldb)
```

$\text{fconvert } A \leftarrow B \bmod F.$

## Parameters

|       |                        |
|-------|------------------------|
| $F$   | field                  |
| $m$   | number of rows         |
| $n$   | number of cols         |
| $A$   | matrix in OtherElement |
| $lda$ | stride of $A$          |
| $B$   | matrix in $F$          |
| $ldb$ | stride of $B$          |

**Todo** check if  $n == lda$

**11.1.3.171 fnegin()** [2/4]

```
template<class Field >
void fnegin (
 const Field & F,
 const size_t m,
 const size_t n,
 typename Field::Element_ptr A,
 const size_t lda)
```

$\text{fnegin } A \leftarrow -A.$

## Parameters

|       |                |
|-------|----------------|
| $F$   | field          |
| $m$   | number of rows |
| $n$   | number of cols |
| $A$   | matrix in $F$  |
| $lda$ | stride of $A$  |

**Todo** check if  $n == lda$

### 11.1.3.172 fneg() [2/4]

```
template<class Field >
void fneg (
 const Field & F,
 const size_t m,
 const size_t n,
 typename Field::ConstElement_ptr B,
 const size_t ldb,
 typename Field::Element_ptr A,
 const size_t lda)
```

$\text{fneg } A \leftarrow -B.$

#### Parameters

|       |                |
|-------|----------------|
| $F$   | field          |
| $m$   | number of rows |
| $n$   | number of cols |
| $A$   | matrix in $F$  |
| $lda$ | stride of $A$  |

**Todo** check if  $n == lda$

### 11.1.3.173 faxpby() [2/2]

```
template<class Field >
void faxpby (
 const Field & F,
 const size_t m,
 const size_t n,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr X,
 const size_t ldx,
 const typename Field::Element beta,
 typename Field::Element_ptr Y,
 const size_t ldy)
```

$\text{faxpby} : y \leftarrow \alpha \cdot x + \beta \cdot y.$

#### Parameters

|    |          |                          |
|----|----------|--------------------------|
|    | $F$      | field                    |
|    | $m$      | row dimension            |
|    | $n$      | column dimension         |
|    | $\alpha$ | scalar                   |
| in | $X$      | vector in $F$            |
|    | $ldx$    | leading dimension of $X$ |
|    | $\beta$  | scalar                   |
|    | $ldy$    | leading dimension of $Y$ |

**Note**

this is a catlas function

**11.1.3.174 fmove()** [1/2]

```
template<class Field >
void fmove (
 const Field & F,
 const size_t m,
 const size_t n,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr B,
 const size_t ldb)
```

$\text{fmove} : A \leftarrow B \text{ and } B \leftarrow 0.$

**Parameters**

|       |                        |
|-------|------------------------|
| $F$   | field                  |
| $m$   | number of rows to copy |
| $n$   | number of cols to copy |
| $A$   | matrix in $F$          |
| $lda$ | stride of $A$          |
| $B$   | matrix in $F$          |
| $ldb$ | stride of $B$          |

**11.1.3.175 bitsize()**

```
template<class Field >
size_t bitsize (
 const Field & F,
 size_t M,
 size_t N,
 const typename Field::ConstElement_ptr A,
 size_t lda) [inline]
```

bitsize: Computes the largest bitsize of the matrix' coefficients.

If the matrix is over a modular prime field, it returns the bitsize of the largest element (in a bsolute value)

**Parameters**

|        |                                                           |
|--------|-----------------------------------------------------------|
| $F$    | field                                                     |
| $M$    | rows                                                      |
| $N$    | cols                                                      |
| $incX$ | stride of $X$                                             |
| $A$    | a matrix of leading dimension $lda$ and size $M \times N$ |
| $lda$  | leading dimension of $A$                                  |

**11.1.3.176** `bitsize< Givaro::ZRing< Givaro::Integer > >()`

```
template<>
size_t bitsize< Givaro::ZRing< Givaro::Integer > > (
 const Givaro::ZRing< Givaro::Integer > & F,
 size_t M,
 size_t N,
 const Givaro::Integer * A,
 size_t lda) [inline]
```

**11.1.3.177** `ftrmv()`

```
template<class Field >
void ftrmv (
 const Field & F,
 const FFLAS_UPLO Uplo,
 const FFLAS_TRANSPOSE TransA,
 const FFLAS_DIAG Diag,
 const size_t N,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::Element_ptr X,
 int incX)
```

ftrsm: TRIangular Matrix Vector prodcut Computes  $X \leftarrow \text{op}(A)X$

**Parameters**

|               |                                                   |
|---------------|---------------------------------------------------|
| <i>F</i>      | field                                             |
| <i>X</i>      | vector of size N on a field F                     |
| <i>incX</i>   | stride of X                                       |
| <i>A</i>      | a matrix of leading dimension lda and size N      |
| <i>lda</i>    | leading dimension of A                            |
| <i>N</i>      | number of rows and columns of A                   |
| <i>TransA</i> | if TransA==FflasTrans then $\text{op}(A) = A^T$ . |
| <i>Diag</i>   | if Diag==FflasUnit then A is unit diagonal.       |
| <i>Uplo</i>   | if Uplo==FflasUpper then A is upper triangular    |

**11.1.3.178** `ftrsm()` [6/9]

```
template<class Field >
void ftrsm (
 const Field & F,
 const FFLAS_SIDE Side,
 const FFLAS_UPLO Uplo,
 const FFLAS_TRANSPOSE TransA,
 const FFLAS_DIAG Diag,
 const size_t M,
 const size_t N,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
```

```

 const size_t lda,
 typename Field::Element_ptr B,
 const size_t ldb)

```

ftsrsm: **TR**angular **S**ystem solve with **M**atrix.

Computes  $B \leftarrow \alpha \text{op}(A^{-1})B$  or  $B \leftarrow \alpha B \text{op}(A^{-1})$ .

#### Parameters

|               |                                                                                                       |
|---------------|-------------------------------------------------------------------------------------------------------|
| <i>F</i>      | field                                                                                                 |
| <i>Side</i>   | if Side==FflasLeft then $B \leftarrow \alpha \text{op}(A^{-1})B$ is computed.                         |
| <i>Uplo</i>   | if Uplo==FflasUpper then A is upper triangular                                                        |
| <i>TransA</i> | if TransA==FflasTrans then $\text{op}(A) = A^t$ .                                                     |
| <i>Diag</i>   | if Diag==FflasUnit then A is unit.                                                                    |
| <i>M</i>      | rows of B                                                                                             |
| <i>N</i>      | cols of B                                                                                             |
| <i>alpha</i>  | scalar                                                                                                |
| <i>A</i>      | triangular invertible matrix. If Side==FflasLeft then A is $N \times N$ , otherwise A is $M \times M$ |
| <i>lda</i>    | leading dim of A                                                                                      |
| <i>B</i>      | matrix of size MxN                                                                                    |
| <i>ldb</i>    | leading dim of B                                                                                      |

**Bug**  $\alpha$  must be non zero.

#### 11.1.3.179 fsyrk\_strassen() [2/2]

```

template<class Field , typename FieldTrait >
Field::Element_ptr fsyrk_strassen (
 const Field & F,
 const FFLAS_UPLO UpLo,
 const FFLAS_TRANSPOSE trans,
 const size_t N,
 const size_t K,
 const typename Field::Element y1,
 const typename Field::Element y2,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > & H) [inline]

```

#### 11.1.3.180 pfgemm() [1/7]

```

template<typename Field >
Field::Element_ptr pfgemm (
 const Field & F,

```

```

const FFLAS_TRANSPOSE ta,
const FFLAS_TRANSPOSE tb,
const size_t m,
const size_t n,
const size_t k,
const typename Field::Element alpha,
typename Field::ConstElement_ptr A,
const size_t lda,
typename Field::ConstElement_ptr B,
const size_t ldb,
const typename Field::Element beta,
typename Field::Element_ptr C,
const size_t ldc,
size_t numthreads = 0)

```

#### 11.1.3.181 pfgemm\_1D\_rec()

```

template<class Field >
Field::Element * pfgemm_1D_rec (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const typename Field::Element alpha,
 const typename Field::Element_ptr A,
 const size_t lda,
 const typename Field::Element_ptr B,
 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element * C,
 const size_t ldc,
 size_t seuil)

```

#### 11.1.3.182 pfgemm\_2D\_rec()

```

template<class Field >
Field::Element * pfgemm_2D_rec (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const typename Field::Element alpha,
 const typename Field::Element_ptr A,
 const size_t lda,
 const typename Field::Element_ptr B,
 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element * C,
 const size_t ldc,
 size_t seuil)

```

**11.1.3.183 pfgemm\_3D\_rec()**

```

template<class Field >
Field::Element * pfgemm_3D_rec (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const typename Field::Element alpha,
 const typename Field::Element_ptr A,
 const size_t lda,
 const typename Field::Element_ptr B,
 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 size_t seuil,
 size_t * x)

```

**11.1.3.184 pfgemm\_3D\_rec2()**

```

template<class Field >
Field::Element_ptr pfgemm_3D_rec2 (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const typename Field::Element alpha,
 const typename Field::Element_ptr A,
 const size_t lda,
 const typename Field::Element_ptr B,
 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 size_t seuil,
 size_t * x)

```

**11.1.3.185 fgemm() [19/23]**

```

template<class Field , class ModeTrait , class Strat , class Param >
std::enable_if<!std::is_same< ModeTrait, ModeCategories::ConvertTo< ElementCategories::RNSElementTag
> >::value, typename Field::Element_ptr >::type fgemm (
 const Field & F,
 const FFLAS::FFLAS_TRANSPOSE ta,
 const FFLAS::FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const typename Field::Element alpha,

```



```

 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr B,
 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 MMHelper< Field, MMHelperAlgo::Winograd, ModeTrait, ParSeqHelper::Parallel< Strat,
Param > > & H) [inline]

```

#### 11.1.3.186 ftrsm() [7/9]

```

template<class Field , class Cut , class Param >
Field::Element_ptr ftrsm (
 const Field & F,
 const FFLAS::FFLAS_SIDE Side,
 const FFLAS::FFLAS_UPLO UpLo,
 const FFLAS::FFLAS_TRANSPOSE TA,
 const FFLAS::FFLAS_DIAG Diag,
 const size_t m,
 const size_t n,
 const typename Field::Element alpha,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr B,
 const size_t ldb,
 TRSMHelper< StructureHelper::Iterative, ParSeqHelper::Parallel< Cut, Param > > &
H) [inline]

```

#### 11.1.3.187 ftrsm() [8/9]

```

template<class Field , class Cut , class Param >
Field::Element_ptr ftrsm (
 const Field & F,
 const FFLAS::FFLAS_SIDE Side,
 const FFLAS::FFLAS_UPLO UpLo,
 const FFLAS::FFLAS_TRANSPOSE TA,
 const FFLAS::FFLAS_DIAG Diag,
 const size_t m,
 const size_t n,
 const typename Field::Element alpha,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr B,
 const size_t ldb,
 TRSMHelper< StructureHelper::Hybrid, ParSeqHelper::Parallel< Cut, Param > > & H
) [inline]

```

#### 11.1.3.188 fspmv() [1/2]

```

template<class Field , class SM >
void fspmv (
 const Field & F,
 const SM & A,
 typename Field::ConstElement_ptr x,
 const typename Field::Element & beta,
 typename Field::Element_ptr y) [inline]

```

**11.1.3.189 fspmm()**

```
template<class Field , class SM >
void fspmm (
 const Field & F,
 const SM & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 const typename Field::Element & beta,
 typename Field::Element_ptr y,
 int ldy) [inline]
```

**11.1.3.190 sparse\_init() [1/16]**

```
template<class Field , class IndexT >
void sparse_init (
 const Field & F,
 Sparse< Field, SparseMatrix_t::COO > & A,
 const IndexT * row,
 const IndexT * col,
 typename Field::ConstElement_ptr dat,
 uint64_t rowdim,
 uint64_t coldim,
 uint64_t nnz) [inline]
```

**11.1.3.191 sparse\_init() [2/16]**

```
template<class Field , class IndexT >
void sparse_init (
 const Field & F,
 Sparse< Field, SparseMatrix_t::COO_ZO > & A,
 const IndexT * row,
 const IndexT * col,
 typename Field::ConstElement_ptr dat,
 uint64_t rowdim,
 uint64_t coldim,
 uint64_t nnz) [inline]
```

**11.1.3.192 sparse\_delete() [1/12]**

```
template<class Field >
void sparse_delete (
 const Sparse< Field, SparseMatrix_t::COO > & A) [inline]
```

**11.1.3.193 sparse\_delete() [2/12]**

```
template<class Field >
void sparse_delete (
 const Sparse< Field, SparseMatrix_t::COO_ZO > & A) [inline]
```

**11.1.3.194 sparse\_init()** [3/16]

```
template<class Field , class IndexT >
void sparse_init (
 const Field & F,
 Sparse< Field, SparseMatrix_t::CSR > & A,
 const IndexT * row,
 const IndexT * col,
 typename Field::ConstElement_ptr dat,
 uint64_t rowdim,
 uint64_t coldim,
 uint64_t nnz) [inline]
```

**11.1.3.195 sparse\_init()** [4/16]

```
template<class Field , class IndexT >
void sparse_init (
 const Field & F,
 Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
 const IndexT * row,
 const IndexT * col,
 typename Field::ConstElement_ptr dat,
 uint64_t rowdim,
 uint64_t coldim,
 uint64_t nnz) [inline]
```

**11.1.3.196 sparse\_delete()** [3/12]

```
template<class Field >
void sparse_delete (
 const Sparse< Field, SparseMatrix_t::CSR > & A) [inline]
```

**11.1.3.197 sparse\_delete()** [4/12]

```
template<class Field >
void sparse_delete (
 const Sparse< Field, SparseMatrix_t::CSR_ZO > & A) [inline]
```

**11.1.3.198 sparse\_print()** [1/3]

```
template<class Field >
std::ostream & sparse_print (
 std::ostream & os,
 const Sparse< Field, SparseMatrix_t::CSR > & A) [inline]
```

**11.1.3.199 sparse\_init() [5/16]**

```
template<class IndexT >
void sparse_init (
 const Givaro::Modular< Givaro::Integer > & F,
 Sparse< Givaro::Modular< Givaro::Integer >, SparseMatrix_t::CSR > & A,
 const IndexT * row,
 const IndexT * col,
 Givaro::Integer * dat,
 uint64_t rowdim,
 uint64_t coldim,
 uint64_t nnz) [inline]
```

**11.1.3.200 sparse\_init() [6/16]**

```
template<class IndexT >
void sparse_init (
 const Givaro::ZRing< Givaro::Integer > & F,
 Sparse< Givaro::ZRing< Givaro::Integer >, SparseMatrix_t::CSR_ZO > & A,
 const IndexT * row,
 const IndexT * col,
 Givaro::Integer * dat,
 uint64_t rowdim,
 uint64_t coldim,
 uint64_t nnz) [inline]
```

**11.1.3.201 sparse\_init() [7/16]**

```
template<class IndexT , size_t RECINT_SIZE>
void sparse_init (
 const Givaro::ZRing< RecInt::rmint< RECINT_SIZE > > & F,
 Sparse< Givaro::ZRing< RecInt::rmint< RECINT_SIZE > >, SparseMatrix_t::CSR_ZO >
& A,
 const IndexT * row,
 const IndexT * col,
 typename Givaro::ZRing< RecInt::rmint< RECINT_SIZE > >::Element_ptr dat,
 uint64_t rowdim,
 uint64_t coldim,
 uint64_t nnz) [inline]
```

**11.1.3.202 sparse\_init() [8/16]**

```
template<class IndexT , size_t RECINT_SIZE>
void sparse_init (
 const Givaro::ZRing< RecInt::rmint< RECINT_SIZE > > & F,
 Sparse< Givaro::ZRing< RecInt::rmint< RECINT_SIZE > >, SparseMatrix_t::CSR > &
A,
 const IndexT * row,
 const IndexT * col,
 typename Givaro::ZRing< RecInt::rmint< RECINT_SIZE > >::Element_ptr dat,
 uint64_t rowdim,
 uint64_t coldim,
 uint64_t nnz) [inline]
```

**11.1.3.203 sparse\_delete()** [5/12]

```
template<class Field >
void sparse_delete (
 const Sparse< Field, SparseMatrix_t::CSR_HYB > & A) [inline]
```

**11.1.3.204 sparse\_init()** [9/16]

```
template<class Field , class IndexT >
void sparse_init (
 const Field & F,
 Sparse< Field, SparseMatrix_t::CSR_HYB > & A,
 const IndexT * row,
 const IndexT * col,
 typename Field::ConstElement_ptr dat,
 uint64_t rowdim,
 uint64_t coldim,
 uint64_t nnz) [inline]
```

**11.1.3.205 sparse\_init()** [10/16]

```
template<class Field , class IndexT >
void sparse_init (
 const Field & F,
 Sparse< Field, SparseMatrix_t::ELL > & A,
 const IndexT * row,
 const IndexT * col,
 typename Field::ConstElement_ptr dat,
 uint64_t rowdim,
 uint64_t coldim,
 uint64_t nnz) [inline]
```

**11.1.3.206 sparse\_init()** [11/16]

```
template<class Field , class IndexT >
void sparse_init (
 const Field & F,
 Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
 const IndexT * row,
 const IndexT * col,
 typename Field::ConstElement_ptr dat,
 uint64_t rowdim,
 uint64_t coldim,
 uint64_t nnz) [inline]
```

**11.1.3.207 sparse\_delete()** [6/12]

```
template<class Field >
void sparse_delete (
 const Sparse< Field, SparseMatrix_t::ELL > & A) [inline]
```

**11.1.3.208 sparse\_delete()** [7/12]

```
template<class Field >
void sparse_delete (
 const Sparse< Field, SparseMatrix_t::ELL_ZO > & A) [inline]
```

**11.1.3.209 sparse\_init()** [12/16]

```
template<class Field , class IndexT >
void sparse_init (
 const Field & F,
 Sparse< Field, SparseMatrix_t::ELL_simd > & A,
 const IndexT * row,
 const IndexT * col,
 typename Field::ConstElement_ptr dat,
 uint64_t rowdim,
 uint64_t coldim,
 uint64_t nnz) [inline]
```

**11.1.3.210 sparse\_init()** [13/16]

```
template<class Field , class IndexT >
void sparse_init (
 const Field & F,
 Sparse< Field, SparseMatrix_t::ELL_simd_ZO > & A,
 const IndexT * row,
 const IndexT * col,
 typename Field::ConstElement_ptr dat,
 uint64_t rowdim,
 uint64_t coldim,
 uint64_t nnz) [inline]
```

**11.1.3.211 sparse\_delete()** [8/12]

```
template<class Field >
void sparse_delete (
 const Sparse< Field, SparseMatrix_t::ELL_simd > & A) [inline]
```

**11.1.3.212 sparse\_delete()** [9/12]

```
template<class Field >
void sparse_delete (
 const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > & A) [inline]
```

**11.1.3.213 sparse\_print()** [2/3]

```
template<class Field >
void sparse_print (
 const Sparse< Field, SparseMatrix_t::ELL_simd > & A) [inline]
```

**11.1.3.214 sparse\_delete()** [10/12]

```
template<class Field >
void sparse_delete (
 const Sparse< Field, SparseMatrix_t::HYB_ZO > & A) [inline]
```

**11.1.3.215 sparse\_init()** [14/16]

```
template<class Field , class IndexT >
void sparse_init (
 const Field & F,
 Sparse< Field, SparseMatrix_t::HYB_ZO > & A,
 const IndexT * row,
 const IndexT * col,
 typename Field::ConstElement_ptr dat,
 uint64_t rowdim,
 uint64_t coldim,
 uint64_t nnz) [inline]
```

**11.1.3.216 operator<<()**

```
template<typename _Field >
std::ostream & operator<< (
 std::ostream & os,
 const Sparse< _Field, SparseMatrix_t::HYB_ZO > & A)
```

**11.1.3.217 readSmsFormat()**

```
template<class Field , bool sorted = true, bool read_integer = false>
void readSmsFormat (
 const std::string & path,
 const Field & f,
 index_t *& row,
 index_t *& col,
 typename Field::Element_ptr & val,
 index_t & rowdim,
 index_t & coldim,
 uint64_t & nnz)
```

**11.1.3.218 readSprFormat()**

```
template<class Field >
void readSprFormat (
 const std::string & path,
 const Field & f,
 index_t *& row,
 index_t *& col,
 typename Field::Element_ptr & val,
 index_t & rowdim,
 index_t & coldim,
 uint64_t & nnz)
```

**11.1.3.219** `getDataType()` [1/4]

```
template<class T >
std::enable_if< std::is_integral< T >::value, int > getDataType ()
```

**11.1.3.220** `getDataType()` [2/4]

```
template<class T >
std::enable_if< std::is_floating_point< T >::value, int > getDataType ()
```

**11.1.3.221** `getDataType()` [3/4]

```
template<class T >
std::enable_if< std::is_same< T, mpz_t >::value, int > getDataType ()
```

**11.1.3.222** `getDataType()` [4/4]

```
template<class T >
int getDataType ()
```

**11.1.3.223** `readMachineType()`

```
template<class Field >
void readMachineType (
 const Field & F,
 typename Field::Element & modulo,
 typename Field::Element_ptr val,
 std::ifstream & file,
 const uint64_t dims,
 const mask_t data_type,
 const mask_t field_desc)
```

**11.1.3.224** `readDnsFormat()`

```
template<class Field >
void readDnsFormat (
 const std::string & path,
 const Field & F,
 index_t & rowdim,
 index_t & coldim,
 typename Field::Element_ptr & val)
```

**11.1.3.225** `writeDnsFormat()`

```
template<class Field >
void writeDnsFormat (
 const std::string & path,
 const Field & F,
 const index_t & rowdim,
 const index_t & coldim,
 typename Field::Element_ptr A,
 index_t ldA)
```



**11.1.3.226 fspmv()** [2/2]

```
template<class Field >
void fspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::SELL_ZO > & A,
 typename Field::ConstElement_ptr x,
 typename Field::Element_ptr y,
 FieldCategories::ModularTag) [inline]
```

**11.1.3.227 sparse\_delete()** [11/12]

```
template<class Field >
void sparse_delete (
 const Sparse< Field, SparseMatrix_t::SELL > & A) [inline]
```

**11.1.3.228 sparse\_delete()** [12/12]

```
template<class Field >
void sparse_delete (
 const Sparse< Field, SparseMatrix_t::SELL_ZO > & A) [inline]
```

**11.1.3.229 sparse\_print()** [3/3]

```
template<class Field >
void sparse_print (
 const Sparse< Field, SparseMatrix_t::SELL > & A) [inline]
```

**11.1.3.230 sparse\_init()** [15/16]

```
template<class Field , class IndexT >
void sparse_init (
 const Field & F,
 Sparse< Field, SparseMatrix_t::SELL > & A,
 const IndexT * row,
 const IndexT * col,
 typename Field::ConstElement_ptr dat,
 uint64_t rowdim,
 uint64_t coldim,
 uint64_t nnz,
 uint64_t sigma = 0) [inline]
```

**11.1.3.231 sparse\_init()** [16/16]

```
template<class Field , class IndexT >
void sparse_init (
 const Field & F,
 Sparse< Field, SparseMatrix_t::SELL_ZO > & A,
 const IndexT * row,
 const IndexT * col,
 typename Field::ConstElement_ptr dat,
 uint64_t rowdim,
 uint64_t coldim,
 uint64_t nnz) [inline]
```

**11.1.3.232 computeDeviation()**

```
template<class It >
double computeDeviation (
 It begin,
 It end)
```

**11.1.3.233 getStat()**

```
template<class Field >
StatsMatrix getStat (
 const Field & F,
 const index_t * row,
 const index_t * col,
 typename Field::ConstElement_ptr val,
 uint64_t rowdim,
 uint64_t coldim,
 uint64_t nnz)
```

**11.1.3.234 maxCardinality()**

```
template<class Field , class enable = void>
Field::Residu_t maxCardinality () [inline]
```

**11.1.3.235 maxCardinality< Givaro::Modular< int64\_t > >()**

```
template<>
uint64_t maxCardinality< Givaro::Modular< int64_t > > () [inline]
```

**11.1.3.236 maxCardinality< Givaro::Modular< int32\_t > >()**

```
template<>
uint32_t maxCardinality< Givaro::Modular< int32_t > > () [inline]
```

**11.1.3.237 minCardinality()**

```
template<class Field >
Field::Residu_t minCardinality () [inline]
```

**11.1.3.238 fflas\_delete() [1/4]**

```
template<>
void fflas_delete (
 FFPACK::rns_double_elt_ptr A) [inline]
```

**11.1.3.239 fflas\_delete()** [2/4]

```
template<>
void fflas_delete (
 FFPACK::rns_double_elt_cstptr A) [inline]
```

**11.1.3.240 fflas\_new()** [1/7]

```
template<>
FFPACK::rns_double_elt_ptr fflas_new (
 const FFPACK::RNSIntegerMod< FFPACK::rns_double > & F,
 const size_t m,
 const Alignment align) [inline]
```

**11.1.3.241 fflas\_new()** [2/7]

```
template<>
FFPACK::rns_double_elt_ptr fflas_new (
 const FFPACK::RNSIntegerMod< FFPACK::rns_double > & F,
 const size_t m,
 const size_t n,
 const Alignment align) [inline]
```

**11.1.3.242 finit\_rns()** [1/2]

```
template<typename RNS >
void finit_rns (
 const FFPACK::RNSIntegerMod< RNS > & F,
 const size_t m,
 const size_t n,
 size_t k,
 const Givaro::Integer * B,
 const size_t ldb,
 typename RNS::Element_ptr A)
```

**11.1.3.243 finit\_trans\_rns()**

```
template<typename RNS >
void finit_trans_rns (
 const FFPACK::RNSIntegerMod< RNS > & F,
 const size_t m,
 const size_t n,
 size_t k,
 const Givaro::Integer * B,
 const size_t ldb,
 typename RNS::Element_ptr A)
```

**11.1.3.244 fconvert\_rns()** [1/2]

```

template<typename RNS >
void fconvert_rns (
 const FFPACK::RNSIntegerMod< RNS > & F,
 const size_t m,
 const size_t n,
 Givaro::Integer alpha,
 Givaro::Integer * B,
 const size_t ldb,
 typename RNS::ConstElement_ptr A)

```

**11.1.3.245 fconvert\_trans\_rns()**

```

template<typename RNS >
void fconvert_trans_rns (
 const FFPACK::RNSIntegerMod< RNS > & F,
 const size_t m,
 const size_t n,
 Givaro::Integer alpha,
 Givaro::Integer * B,
 const size_t ldb,
 typename RNS::ConstElement_ptr A)

```

**11.1.3.246 fflas\_new()** [3/7]

```

template<>
FFPACK::rns_double_elt_ptr fflas_new (
 const FFPACK::RNSInteger< FFPACK::rns_double > & F,
 const size_t m,
 const Alignment align) [inline]

```

**11.1.3.247 fflas\_new()** [4/7]

```

template<>
FFPACK::rns_double_elt_ptr fflas_new (
 const FFPACK::RNSInteger< FFPACK::rns_double > & F,
 const size_t m,
 const size_t n,
 const Alignment align) [inline]

```

**11.1.3.248 finit\_rns()** [2/2]

```

template<typename RNS >
void finit_rns (
 const FFPACK::RNSInteger< RNS > & F,
 const size_t m,
 const size_t n,
 size_t k,
 const Givaro::Integer * B,
 const size_t ldb,
 typename FFPACK::RNSInteger< RNS >::Element_ptr A)

```

**11.1.3.249 fconvert\_rns()** [2/2]

```

template<typename RNS >
void fconvert_rns (
 const FFPACK::RNSInteger< RNS > & F,
 const size_t m,
 const size_t n,
 Givaro::Integer alpha,
 Givaro::Integer * B,
 const size_t ldb,
 typename FFPACK::RNSInteger< RNS >::ConstElement_ptr A)

```

**11.1.3.250 freduce()** [8/11]

```

template INST_OR_DECL void freduce (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t n,
 FFLAS_ELT * X,
 const size_t incX)

```

freduce  $x \leftarrow x \bmod F$ .

**Parameters**

|        |                     |
|--------|---------------------|
| $F$    | field               |
| $n$    | size of the vectors |
| $X$    | vector in $F$       |
| $incX$ | stride of $X$       |

**Bug** use cblas\_(d)scal when possible

**11.1.3.251 freduce()** [9/11]

```

template INST_OR_DECL void freduce (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t n,
 const FFLAS_ELT * Y,
 const size_t incY,
 FFLAS_ELT * X,
 const size_t incX)

```

freduce  $x \leftarrow y \bmod F$ .

**Parameters**

|        |                     |
|--------|---------------------|
| $F$    | field               |
| $n$    | size of the vectors |
| $Y$    | vector of Element   |
| $incY$ | stride of $Y$       |
| $X$    | vector in $F$       |
| $incX$ | stride of $X$       |

**Bug** use `cblas_(d)scal` when possible

#### 11.1.3.252 `finit()` [7/8]

```
template INST_OR_DECL void finit (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t n,
 const FFLAS_ELT * Y,
 const size_t incY,
 FFLAS_ELT * X,
 const size_t incX)
```

$\text{finit } x \leftarrow y \bmod F.$

##### Parameters

|        |                                     |
|--------|-------------------------------------|
| $F$    | field                               |
| $n$    | size of the vectors                 |
| $Y$    | vector of <code>OtherElement</code> |
| $incY$ | stride of $Y$                       |
| $X$    | vector in $F$                       |
| $incX$ | stride of $X$                       |

**Bug** use `cblas_(d)scal` when possible

#### 11.1.3.253 `fconvert()` [3/3]

```
template INST_OR_DECL void fconvert (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t n,
 FFLAS_ELT * X,
 const size_t incX,
 const FFLAS_ELT * Y,
 const size_t incY)
```

$\text{fconvert } x \leftarrow y \bmod F.$

##### Parameters

|        |                                     |
|--------|-------------------------------------|
| $F$    | field                               |
| $n$    | size of the vectors                 |
| $Y$    | vector of $F$                       |
| $incY$ | stride of $Y$                       |
| $X$    | vector in <code>OtherElement</code> |
| $incX$ | stride of $X$                       |

**Bug** use `cblas_(d)scal` when possible

**11.1.3.254 fnegin()** [3/4]

```
template INST_OR_DECL void fnegin (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t n,
 FFLAS_ELT * X,
 const size_t incX)
```

$\text{fnegin } x \leftarrow -x.$

**Parameters**

|               |                     |
|---------------|---------------------|
| $F$           | field               |
| $n$           | size of the vectors |
| $X$           | vector in $F$       |
| $\text{incX}$ | stride of $X$       |

**Bug** use `cblas_(d)scal` when possible

**11.1.3.255 fneg()** [3/4]

```
template INST_OR_DECL void fneg (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t n,
 const FFLAS_ELT * Y,
 const size_t incY,
 FFLAS_ELT * X,
 const size_t incX)
```

$\text{fneg } x \leftarrow -y.$

**Parameters**

|               |                     |
|---------------|---------------------|
| $F$           | field               |
| $n$           | size of the vectors |
| $X$           | vector in $F$       |
| $\text{incX}$ | stride of $X$       |
| $Y$           | vector in $F$       |
| $\text{incY}$ | stride of $Y$       |

**Bug** use `cblas_(d)scal` when possible

**11.1.3.256 fzero()** [4/5]

```
template INST_OR_DECL void fzero (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t n,
```

```

FFLAS_ELT * X,
const size_t incX)

```

fzero :  $A \leftarrow 0$ .

#### Parameters

|        |                            |
|--------|----------------------------|
| $F$    | field                      |
| $n$    | number of elements to zero |
| $X$    | vector in $F$              |
| $incX$ | stride of $X$              |

#### 11.1.3.257 fiszero() [3/4]

```

template INST_OR_DECL bool fiszero (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t n,
 const FFLAS_ELT * X,
 const size_t incX)

```

fiszero : test  $X = 0$ .

#### Parameters

|        |                  |
|--------|------------------|
| $F$    | field            |
| $n$    | vector dimension |
| $X$    | vector in $F$    |
| $incX$ | increment of $X$ |

#### 11.1.3.258 fequal() [3/4]

```

template INST_OR_DECL bool fequal (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t n,
 const FFLAS_ELT * X,
 const size_t incX,
 const FFLAS_ELT * Y,
 const size_t incY)

```

fequal : test  $X = Y$ .

#### Parameters

|        |                  |
|--------|------------------|
| $F$    | field            |
| $n$    | vector dimension |
| $X$    | vector in $F$    |
| $incX$ | increment of $X$ |
| $Y$    | vector in $F$    |
| $incY$ | increment of $Y$ |



**11.1.3.259 fassign()** [9/10]

```
template INST_OR_DECL void fassign (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t N,
 const FFLAS_ELT * Y,
 const size_t incY,
 FFLAS_ELT * X,
 const size_t incX)
```

fassign :  $x \leftarrow y$ .

X is preallocated

**Todo** variant for triangular matrix

**Parameters**

|     |        |                     |
|-----|--------|---------------------|
|     | $F$    | field               |
|     | $N$    | size of the vectors |
| out | $X$    | vector in $F$       |
|     | $incX$ | stride of X         |
| in  | $Y$    | vector in $F$       |
|     | $incY$ | stride of Y         |

**11.1.3.260 fscaln()** [9/10]

```
template INST_OR_DECL void fscaln (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t n,
 const FFLAS_ELT alpha,
 FFLAS_ELT * X,
 const size_t incX)
```

fscaln  $x \leftarrow \alpha \cdot x$ .

**Parameters**

|         |                     |
|---------|---------------------|
| $F$     | field               |
| $n$     | size of the vectors |
| $alpha$ | scalar              |
| $X$     | vector in $F$       |
| $incX$  | stride of X         |

**Bug** use cblas\_(d)scal when possible

**Todo** check if comparison with +/-1,0 is necessary.

**11.1.3.261 fscal()** [9/10]

```
template INST_OR_DECL void fscal (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t n,
 const FFLAS_ELT alpha,
 const FFLAS_ELT * X,
 const size_t incX,
 FFLAS_ELT * Y,
 const size_t incY)
```

$\text{fscal } y \leftarrow \alpha \cdot x.$

**Parameters**

|     |               |                     |
|-----|---------------|---------------------|
|     | $F$           | field               |
|     | $n$           | size of the vectors |
|     | $\alpha$      | scalar              |
| in  | $X$           | vector in F         |
|     | $\text{incX}$ | stride of X         |
| out | $Y$           | vector in F         |
|     | $\text{incY}$ | stride of Y         |

**Bug** use cblas\_(d)scal when possible

**Todo** check if comparison with +/-1,0 is necessary.

**11.1.3.262 faxpy()** [5/6]

```
template INST_OR_DECL void faxpy (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t N,
 const FFLAS_ELT alpha,
 const FFLAS_ELT * X,
 const size_t incX,
 FFLAS_ELT * Y,
 const size_t incY)
```

$\text{faxpy} : y \leftarrow \alpha \cdot x + y.$

**Parameters**

|         |               |                     |
|---------|---------------|---------------------|
|         | $F$           | field               |
|         | $N$           | size of the vectors |
|         | $\alpha$      | scalar              |
| in      | $X$           | vector in F         |
|         | $\text{incX}$ | stride of X         |
| in, out | $Y$           | vector in F         |
|         | $\text{incY}$ | stride of Y         |

**11.1.3.263 fdot()** [10/11]

```
template INST_OR_DECL FFLAS_ELT fdot (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t N,
 const FFLAS_ELT * X,
 const size_t incX,
 const FFLAS_ELT * Y,
 const size_t incY)
```

faxpby:  $y \leftarrow \alpha \cdot x + \beta \cdot y$ .

**Parameters**

|         |          |                     |
|---------|----------|---------------------|
|         | $F$      | field               |
|         | $N$      | size of the vectors |
|         | $\alpha$ | scalar              |
| in      | $X$      | vector in F         |
|         | $incX$   | stride of X         |
|         | $\beta$  | scalar              |
| in, out | $Y$      | vector in F         |
|         | $incY$   | stride of Y         |

**Note**

this is a catlas function

fdot: dot product  $x^T y$ .

**Parameters**

|        |                     |
|--------|---------------------|
| $F$    | field               |
| $N$    | size of the vectors |
| $X$    | vector in F         |
| $incX$ | stride of X         |
| $Y$    | vector in F         |
| $incY$ | stride of Y         |

**11.1.3.264 fswap()** [2/2]

```
template INST_OR_DECL void fswap (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t N,
 FFLAS_ELT * X,
 const size_t incX,
 FFLAS_ELT * Y,
 const size_t incY)
```

fswap:  $X \leftrightarrow Y$ .

**Bug** use cblas\_dswap when double

## Parameters

|        |                     |
|--------|---------------------|
| $F$    | field               |
| $N$    | size of the vectors |
| $X$    | vector in $F$       |
| $incX$ | stride of $X$       |
| $Y$    | vector in $F$       |
| $incY$ | stride of $Y$       |

**11.1.3.265 fadd()** [5/8]

```
template INST_OR_DECL void fadd (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t N,
 const FFLAS_ELT * A,
 const size_t inca,
 const FFLAS_ELT * B,
 const size_t incb,
 FFLAS_ELT * C,
 const size_t incc)
```

**11.1.3.266 fsub()** [3/4]

```
template INST_OR_DECL void fsub (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t N,
 const FFLAS_ELT * A,
 const size_t inca,
 const FFLAS_ELT * B,
 const size_t incb,
 FFLAS_ELT * C,
 const size_t incc)
```

**11.1.3.267 faddin()** [4/5]

```
template INST_OR_DECL void faddin (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t N,
 const FFLAS_ELT * B,
 const size_t incb,
 FFLAS_ELT * C,
 const size_t incc)
```

**11.1.3.268 fadd()** [6/8]

```
template INST_OR_DECL void fadd (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t N,
 const FFLAS_ELT * A,
 const size_t inca,
 const FFLAS_ELT alpha,
 const FFLAS_ELT * B,
 const size_t incb,
 FFLAS_ELT * C,
 const size_t incc)
```

**11.1.3.269 fassign()** [10/10]

```
template INST_OR_DECL void fassign (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t m,
 const size_t n,
 const FFLAS_ELT * B,
 const size_t ldb,
 FFLAS_ELT * A,
 const size_t lda)
```

fassign :  $A \leftarrow B$ .

**Parameters**

|       |                        |
|-------|------------------------|
| $F$   | field                  |
| $m$   | number of rows to copy |
| $n$   | number of cols to copy |
| $A$   | matrix in F            |
| $lda$ | stride of A            |
| $B$   | vector in F            |
| $ldb$ | stride of B            |

**11.1.3.270 fzero()** [5/5]

```
template INST_OR_DECL void fzero (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t m,
 const size_t n,
 FFLAS_ELT * A,
 const size_t lda)
```

fzero :  $A \leftarrow 0$ .

**Parameters**

|       |                        |
|-------|------------------------|
| $F$   | field                  |
| $m$   | number of rows to zero |
| $n$   | number of cols to zero |
| $A$   | matrix in F            |
| $lda$ | stride of A            |

**Warning**

may be buggy if Element is larger than int

**11.1.3.271 fequal()** [4/4]

```
template INST_OR_DECL bool fequal (
 const FFLAS_FIELD< FFLAS_ELT > & F,
```

```

const size_t m,
const size_t n,
const FFLAS_ELT * A,
const size_t lda,
const FFLAS_ELT * B,
const size_t ldb)

```

fequal : test  $A = B$ .

#### Parameters

|       |                        |
|-------|------------------------|
| $F$   | field                  |
| $m$   | row dimension          |
| $n$   | column dimension       |
| $A$   | m x n matrix in $F$    |
| $lda$ | leading dimension of A |
| $B$   | m x n matrix in $F$    |
| $ldb$ | leading dimension of B |

#### 11.1.3.272 fiszero() [4/4]

```

template INST_OR_DECL bool fiszero (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t m,
 const size_t n,
 const FFLAS_ELT * A,
 const size_t lda)

```

fiszero : test  $A = 0$ .

#### Parameters

|       |                        |
|-------|------------------------|
| $F$   | field                  |
| $m$   | row dimension          |
| $n$   | column dimension       |
| $A$   | m x n matrix in $F$    |
| $lda$ | leading dimension of A |

#### 11.1.3.273 fidentity() [3/4]

```

template INST_OR_DECL void fidentity (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t m,
 const size_t n,
 FFLAS_ELT * A,
 const size_t lda,
 const FFLAS_ELT & d)

```

creates a diagonal matrix

**11.1.3.274 fidentity()** [4/4]

```
template INST_OR_DECL void fidentity (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t m,
 const size_t n,
 FFLAS_ELT * A,
 const size_t lda)
```

creates a diagonal matrix

**11.1.3.275 freduce()** [10/11]

```
template INST_OR_DECL void freduce (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t m,
 const size_t n,
 FFLAS_ELT * A,
 const size_t lda)
```

freduce  $A \leftarrow A \bmod F$ .

**Parameters**

|       |                |
|-------|----------------|
| $F$   | field          |
| $m$   | number of rows |
| $n$   | number of cols |
| $A$   | matrix in $F$  |
| $lda$ | stride of $A$  |

**11.1.3.276 freduce()** [11/11]

```
template INST_OR_DECL void freduce (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t m,
 const size_t n,
 const FFLAS_ELT * B,
 const size_t ldb,
 FFLAS_ELT * A,
 const size_t lda)
```

freduce  $A \leftarrow B \bmod F$ .

**Parameters**

|       |                            |
|-------|----------------------------|
| $F$   | field                      |
| $m$   | number of rows             |
| $n$   | number of cols             |
| $A$   | matrix in $F$              |
| $lda$ | stride of $A$              |
| $B$   | matrix in $\text{Element}$ |
| $ldb$ | stride of $B$              |

**11.1.3.277 finit()** [8/8]

```
template INST_OR_DECL void finit (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t m,
 const size_t n,
 const FFLAS_ELT * B,
 const size_t ldb,
 FFLAS_ELT * A,
 const size_t lda)
```

$\text{finit } A \leftarrow B \bmod F.$

**Parameters**

|       |                |
|-------|----------------|
| $F$   | field          |
| $m$   | number of rows |
| $n$   | number of cols |
| $A$   | matrix in F    |
| $lda$ | stride of A    |
| $B$   | matrix in F    |
| $ldb$ | stride of B    |

**11.1.3.278 fnegin()** [4/4]

```
template INST_OR_DECL void fnegin (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t m,
 const size_t n,
 FFLAS_ELT * A,
 const size_t lda)
```

$\text{fnegin } A \leftarrow -A.$

**Parameters**

|       |                |
|-------|----------------|
| $F$   | field          |
| $m$   | number of rows |
| $n$   | number of cols |
| $A$   | matrix in F    |
| $lda$ | stride of A    |

**11.1.3.279 fneg()** [4/4]

```
template INST_OR_DECL void fneg (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t m,
 const size_t n,
 const FFLAS_ELT * B,
 const size_t ldb,
```



```

FFLAS_ELT * A,
const size_t lda)

```

$\text{fneg } A \leftarrow -B.$

#### Parameters

|       |                |
|-------|----------------|
| $F$   | field          |
| $m$   | number of rows |
| $n$   | number of cols |
| $A$   | matrix in $F$  |
| $lda$ | stride of $A$  |

#### 11.1.3.280 fscaln() [10/10]

```

template INST_OR_DECL void fscaln (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t m,
 const size_t n,
 const FFLAS_ELT alpha,
 FFLAS_ELT * A,
 const size_t lda)

```

$\text{fscaln } A \leftarrow a \cdot A.$

#### Parameters

|         |                  |
|---------|------------------|
| $F$     | field            |
| $m$     | number of rows   |
| $n$     | number of cols   |
| $alpha$ | homotecie scalar |
| $A$     | matrix in $F$    |
| $lda$   | stride of $A$    |

#### 11.1.3.281 fsca() [10/10]

```

template INST_OR_DECL void fsca (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t m,
 const size_t n,
 const FFLAS_ELT alpha,
 const FFLAS_ELT * A,
 const size_t lda,
 FFLAS_ELT * B,
 const size_t ldb)

```

$\text{fsca } B \leftarrow a \cdot A.$

#### Parameters

|  |     |       |
|--|-----|-------|
|  | $F$ | field |
|--|-----|-------|

## Parameters

|     |          |                  |
|-----|----------|------------------|
|     | $m$      | number of rows   |
|     | $n$      | number of cols   |
|     | $\alpha$ | homotecie scalar |
| in  | $A$      | matrix in $F$    |
|     | $lda$    | stride of $A$    |
| out | $B$      | matrix in $F$    |
|     | $ldb$    | stride of $B$    |

## 11.1.3.282 faxpy() [6/6]

```
template INST_OR_DECL void faxpy (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t m,
 const size_t n,
 const FFLAS_ELT alpha,
 const FFLAS_ELT * X,
 const size_t ldx,
 FFLAS_ELT * Y,
 const size_t ldy)
```

$\text{faxpy} : y \leftarrow \alpha \cdot x + y.$

## Parameters

|         |          |                          |
|---------|----------|--------------------------|
|         | $F$      | field                    |
|         | $m$      | row dimension            |
|         | $n$      | column dimension         |
|         | $\alpha$ | scalar                   |
| in      | $X$      | vector in $F$            |
|         | $ldx$    | leading dimension of $X$ |
| in, out | $Y$      | vector in $F$            |
|         | $ldy$    | leading dimension of $Y$ |

## 11.1.3.283 fmove() [2/2]

```
template INST_OR_DECL void fmove (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t m,
 const size_t n,
 FFLAS_ELT * A,
 const size_t lda,
 FFLAS_ELT * B,
 const size_t ldb)
```

$\text{faxpby} : y \leftarrow \alpha \cdot x + \beta \cdot y.$

## Parameters

|  |     |       |
|--|-----|-------|
|  | $F$ | field |
|--|-----|-------|

## Parameters

|         |          |                          |
|---------|----------|--------------------------|
|         | $m$      | row dimension            |
|         | $n$      | column dimension         |
|         | $\alpha$ | scalar                   |
| in      | $X$      | vector in $F$            |
|         | $ldx$    | leading dimension of $X$ |
|         | $\beta$  | scalar                   |
| in, out | $Y$      | vector in $F$            |
|         | $ldy$    | leading dimension of $Y$ |

## Note

this is a catlas function

$\text{fmove} : A \leftarrow B \text{ and } B \leftarrow 0.$

## Parameters

|       |                        |
|-------|------------------------|
| $F$   | field                  |
| $m$   | number of rows to copy |
| $n$   | number of cols to copy |
| $A$   | matrix in $F$          |
| $lda$ | stride of $A$          |
| $B$   | vector in $F$          |
| $ldb$ | stride of $B$          |

## 11.1.3.284 fadd() [7/8]

```
template INST_OR_DECL void fadd (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t M,
 const size_t N,
 const FFLAS_ELT * A,
 const size_t lda,
 const FFLAS_ELT * B,
 const size_t ldb,
 FFLAS_ELT * C,
 const size_t ldc)
```

$\text{fadd}$  : matrix addition.

Computes  $C = A + B$ .

## Parameters

|       |                                   |
|-------|-----------------------------------|
| $F$   | field                             |
| $M$   | rows                              |
| $N$   | cols                              |
| $A$   | dense matrix of size $M \times N$ |
| $lda$ | leading dimension of $A$          |

## Parameters

|            |                                   |
|------------|-----------------------------------|
| <i>B</i>   | dense matrix of size $M \times N$ |
| <i>ldb</i> | leading dimension of B            |
| <i>C</i>   | dense matrix of size $M \times N$ |
| <i>ldc</i> | leading dimension of C            |

**11.1.3.285 fsub()** [4/4]

```
template INST_OR_DECL void fsub (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t M,
 const size_t N,
 const FFLAS_ELT * A,
 const size_t lda,
 const FFLAS_ELT * B,
 const size_t ldb,
 FFLAS_ELT * C,
 const size_t ldc)
```

fsub : matrix subtraction.

Computes  $C = A - B$ .

## Parameters

|            |                                   |
|------------|-----------------------------------|
| <i>F</i>   | field                             |
| <i>M</i>   | rows                              |
| <i>N</i>   | cols                              |
| <i>A</i>   | dense matrix of size $M \times N$ |
| <i>lda</i> | leading dimension of A            |
| <i>B</i>   | dense matrix of size $M \times N$ |
| <i>ldb</i> | leading dimension of B            |
| <i>C</i>   | dense matrix of size $M \times N$ |
| <i>ldc</i> | leading dimension of C            |

**11.1.3.286 fsubin()** [3/3]

```
template INST_OR_DECL void fsubin (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t M,
 const size_t N,
 const FFLAS_ELT * B,
 const size_t ldb,
 FFLAS_ELT * C,
 const size_t ldc)
```

fsubin  $C = C - B$

**11.1.3.287 fadd()** [8/8]

```
template INST_OR_DECL void fadd (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t M,
 const size_t N,
 const FFLAS_ELT * A,
 const size_t lda,
 const FFLAS_ELT alpha,
 const FFLAS_ELT * B,
 const size_t ldb,
 FFLAS_ELT * C,
 const size_t ldc)
```

fadd : matrix addition with scaling.

Computes  $C = A + \alpha B$ .

**Parameters**

|              |                          |
|--------------|--------------------------|
| <i>F</i>     | field                    |
| <i>M</i>     | rows                     |
| <i>N</i>     | cols                     |
| <i>A</i>     | dense matrix of size MxN |
| <i>lda</i>   | leading dimension of A   |
| <i>alpha</i> | some scalar              |
| <i>B</i>     | dense matrix of size MxN |
| <i>ldb</i>   | leading dimension of B   |
| <i>C</i>     | dense matrix of size MxN |
| <i>ldc</i>   | leading dimension of C   |

**11.1.3.288 faddin()** [5/5]

```
template INST_OR_DECL void faddin (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t M,
 const size_t N,
 const FFLAS_ELT * B,
 const size_t ldb,
 FFLAS_ELT * C,
 const size_t ldc)
```

faddin

**11.1.3.289 fgemv()** [17/19]

```
template INST_OR_DECL FFLAS_ELT * fgemv (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const FFLAS_TRANSPOSE TransA,
 const size_t M,
 const size_t N,
```

```

const FFLAS_ELT alpha,
const FFLAS_ELT * A,
const size_t lda,
const FFLAS_ELT * X,
const size_t incX,
const FFLAS_ELT beta,
FFLAS_ELT * Y,
const size_t incY)

```

finite prime FFLAS\_FIELD<FFLAS\_ELT> GEneral Matrix Vector multiplication.

Computes  $Y \leftarrow \alpha \text{op}(A)X + \beta Y$ .

#### Parameters

|     |               |                                                   |
|-----|---------------|---------------------------------------------------|
|     | <i>F</i>      | field                                             |
|     | <i>TransA</i> | if TransA==FflasTrans then $\text{op}(A) = A^t$ . |
|     | <i>M</i>      | rows                                              |
|     | <i>N</i>      | cols                                              |
|     | <i>alpha</i>  | scalar                                            |
|     | <i>A</i>      | dense matrix of size MxN                          |
|     | <i>lda</i>    | leading dimension of A                            |
|     | <i>X</i>      | dense vector of size N                            |
|     | <i>incX</i>   | stride of X                                       |
|     | <i>beta</i>   | scalar                                            |
| out | <i>Y</i>      | dense vector of size M                            |
|     | <i>incY</i>   | stride of Y                                       |

#### 11.1.3.290 fger() [12/12]

```

template INST_OR_DECL void fger (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t M,
 const size_t N,
 const FFLAS_ELT alpha,
 const FFLAS_ELT * x,
 const size_t incx,
 const FFLAS_ELT * y,
 const size_t incy,
 FFLAS_ELT * A,
 const size_t lda)

```

fger: rank one update of a general matrix

Computes  $A \leftarrow \alpha x y^T + A$

#### Parameters

|         |              |                                                    |
|---------|--------------|----------------------------------------------------|
|         | <i>F</i>     | field                                              |
|         | <i>M</i>     | rows                                               |
|         | <i>N</i>     | cols                                               |
|         | <i>alpha</i> | scalar                                             |
| in, out | <i>A</i>     | dense matrix of size MxN and leading dimension lda |

## Parameters

|  |             |                        |
|--|-------------|------------------------|
|  | <i>lda</i>  | leading dimension of A |
|  | <i>x</i>    | dense vector of size M |
|  | <i>incx</i> | stride of X            |
|  | <i>y</i>    | dense vector of size N |
|  | <i>incy</i> | stride of Y            |

## 11.1.3.291 ftrsv() [2/2]

```
template INST_OR_DECL void ftrsv (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const FFLAS_UPLO Uplo,
 const FFLAS_TRANSPOSE TransA,
 const FFLAS_DIAG Diag,
 const size_t N,
 const FFLAS_ELT * A,
 const size_t lda,
 FFLAS_ELT * X,
 int incX)
```

ftrsv: TRIangular System solve with Vector Computes  $X \leftarrow \text{op}(A^{-1})X$

## Parameters

|               |                                                    |
|---------------|----------------------------------------------------|
| <i>F</i>      | field                                              |
| <i>X</i>      | vector of size N on a field F                      |
| <i>incX</i>   | stride of X                                        |
| <i>A</i>      | a matrix of leading dimension lda and size N       |
| <i>lda</i>    | leading dimension of A                             |
| <i>N</i>      | number of rows or columns of A according to TransA |
| <i>TransA</i> | if TransA==FflasTrans then $\text{op}(A) = A^t$ .  |
| <i>Diag</i>   | if Diag==FflasUnit then A is unit.                 |
| <i>Uplo</i>   | if Uplo==FflasUpper then A is upper triangular     |

## 11.1.3.292 ftrsm() [9/9]

```
template INST_OR_DECL void ftrsm (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const FFLAS_SIDE Side,
 const FFLAS_UPLO Uplo,
 const FFLAS_TRANSPOSE TransA,
 const FFLAS_DIAG Diag,
 const size_t M,
 const size_t N,
 const FFLAS_ELT alpha,
 const FFLAS_ELT * A,
 const size_t lda,
 FFLAS_ELT * B,
 const size_t ldb)
```

ftfrm: **TR**angular **S**ystem solve with **M**atrix.

Computes  $B \leftarrow \alpha \text{op}(A^{-1})B$  or  $B \leftarrow \alpha B \text{op}(A^{-1})$ .

#### Parameters

|               |                                                                                                       |
|---------------|-------------------------------------------------------------------------------------------------------|
| <i>F</i>      | field                                                                                                 |
| <i>Side</i>   | if Side==FflasLeft then $B \leftarrow \alpha \text{op}(A^{-1})B$ is computed.                         |
| <i>Uplo</i>   | if Uplo==FflasUpper then A is upper triangular                                                        |
| <i>TransA</i> | if TransA==FflasTrans then $\text{op}(A) = A^t$ .                                                     |
| <i>Diag</i>   | if Diag==FflasUnit then A is unit.                                                                    |
| <i>M</i>      | rows of B                                                                                             |
| <i>N</i>      | cols of B                                                                                             |
| <i>alpha</i>  | scalar                                                                                                |
| <i>A</i>      | triangular invertible matrix. If Side==FflasLeft then A is $N \times N$ , otherwise A is $M \times M$ |
| <i>lda</i>    | leading dim of A                                                                                      |
| <i>B</i>      | matrix of size MxN                                                                                    |
| <i>ldb</i>    | leading dim of B                                                                                      |

**Bug**  $\alpha$  must be non zero.

#### 11.1.3.293 ftfrm() [3/3]

```
template INST_OR_DECL void ftfrm (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const FFLAS_SIDE Side,
 const FFLAS_UPLO Uplo,
 const FFLAS_TRANSPOSE TransA,
 const FFLAS_DIAG Diag,
 const size_t M,
 const size_t N,
 const FFLAS_ELT alpha,
 const FFLAS_ELT * A,
 const size_t lda,
 FFLAS_ELT * B,
 const size_t ldb)
```

ftfrm: **TR**angular **M**atrix **M**ultiply.

Computes  $B \leftarrow \alpha \text{op}(A)B$  or  $B \leftarrow \alpha B \text{op}(A)$ .

#### Parameters

|               |                                                                          |
|---------------|--------------------------------------------------------------------------|
| <i>F</i>      | field                                                                    |
| <i>Side</i>   | if Side==FflasLeft then $B \leftarrow \alpha \text{op}(A)B$ is computed. |
| <i>Uplo</i>   | if Uplo==FflasUpper then A is upper triangular                           |
| <i>TransA</i> | if TransA==FflasTrans then $\text{op}(A) = A^t$ .                        |
| <i>Diag</i>   | if Diag==FflasUnit then A is implicitly unit.                            |
| <i>M</i>      | rows of B                                                                |
| <i>N</i>      | cols of B                                                                |



## Parameters

|              |                                                                                                |
|--------------|------------------------------------------------------------------------------------------------|
| <i>alpha</i> | scalar                                                                                         |
| <i>A</i>     | triangular matrix. If Side==FflasLeft then $A$ is $N \times N$ , otherwise $A$ is $M \times M$ |
| <i>lda</i>   | leading dim of A                                                                               |
| <i>B</i>     | matrix of size $M \times N$                                                                    |
| <i>ldb</i>   | leading dim of B                                                                               |

## 11.1.3.294 fgemm() [20/23]

```
template INST_OR_DECL FFLAS_ELT * fgemm (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const FFLAS_ELT alpha,
 const FFLAS_ELT * A,
 const size_t lda,
 const FFLAS_ELT * B,
 const size_t ldb,
 const FFLAS_ELT beta,
 FFLAS_ELT * C,
 const size_t ldc)
```

fgemm: Field **GE**neral **M**atrix **M**ultiply.

Computes  $C = \alpha \text{op}(A) \times \text{op}(B) + \beta C$  Automatically set Winograd recursion level

## Parameters

|              |                                                                                                    |
|--------------|----------------------------------------------------------------------------------------------------|
| <i>F</i>     | field.                                                                                             |
| <i>ta</i>    | if $ta == \text{FflasTrans}$ then $\text{op}(A) = A^t$ , else $\text{op}(A) = A$ ,                 |
| <i>tb</i>    | same for matrix B                                                                                  |
| <i>m</i>     | see A                                                                                              |
| <i>n</i>     | see B                                                                                              |
| <i>k</i>     | see A                                                                                              |
| <i>alpha</i> | scalar                                                                                             |
| <i>beta</i>  | scalar                                                                                             |
| <i>A</i>     | $\text{op}(A)$ is $m \times k$                                                                     |
| <i>B</i>     | $\text{op}(B)$ is $k \times n$                                                                     |
| <i>C</i>     | $C$ is $m \times n$                                                                                |
| <i>lda</i>   | leading dimension of A                                                                             |
| <i>ldb</i>   | leading dimension of B                                                                             |
| <i>ldc</i>   | leading dimension of C                                                                             |
| <i>w</i>     | recursive levels of Winograd's algorithm are used. No argument (or -1) does auto computation of w. |

## Warning

$\alpha$  must be invertible

## 11.1.3.295 fgemm() [21/23]

```
template INST_OR_DECL FFLAS_ELT * fgemm (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const FFLAS_ELT alpha,
 const FFLAS_ELT * A,
 const size_t lda,
 const FFLAS_ELT * B,
 const size_t ldb,
 const FFLAS_ELT beta,
 FFLAS_ELT * C,
 const size_t ldc,
 const ParSeqHelper::Sequential seq)
```

## 11.1.3.296 fgemm() [22/23]

```
template INST_OR_DECL FFLAS_ELT * fgemm (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const FFLAS_ELT alpha,
 const FFLAS_ELT * A,
 const size_t lda,
 const FFLAS_ELT * B,
 const size_t ldb,
 const FFLAS_ELT beta,
 FFLAS_ELT * C,
 const size_t ldc,
 const ParSeqHelper::Parallel< CuttingStrategy::Recursive, StrategyParameter::TwoDAdaptive
> par)
```

## 11.1.3.297 fgemm() [23/23]

```
template INST_OR_DECL FFLAS_ELT * fgemm (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const FFLAS_ELT alpha,
```

```

 const FFLAS_ELT * A,
 const size_t lda,
 const FFLAS_ELT * B,
 const size_t ldb,
 const FFLAS_ELT beta,
 FFLAS_ELT * C,
 const size_t ldc,
 const ParSeqHelper::Parallel< CuttingStrategy::Block, StrategyParameter::Threads
> par)

```

### 11.1.3.298 fsquare() [6/6]

```

template INST_OR_DECL FFLAS_ELT * fsquare (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const FFLAS_TRANSPOSE ta,
 const size_t n,
 const FFLAS_ELT alpha,
 const FFLAS_ELT * A,
 const size_t lda,
 const FFLAS_ELT beta,
 FFLAS_ELT * C,
 const size_t ldc)

```

fsquare: Squares a matrix.

compute  $C \leftarrow \alpha \text{op}(A) \text{op}(A) + \beta C$  over a FFLAS\_FIELD <FFLAS\_ELT>  $\mathbb{F}$  Avoid the conversion of B

#### Parameters

|              |                                                       |
|--------------|-------------------------------------------------------|
| <i>ta</i>    | if $ta == \text{FflasTrans}$ , $\text{op}(A) = A^T$ . |
| <i>F</i>     | field                                                 |
| <i>n</i>     | size of A                                             |
| <i>alpha</i> | scalar                                                |
| <i>beta</i>  | scalar                                                |
| <i>A</i>     | dense matrix of size $n \times n$                     |
| <i>lda</i>   | leading dimension of A                                |
| <i>C</i>     | dense matrix of size $n \times n$                     |
| <i>ldc</i>   | leading dimension of C                                |

### 11.1.3.299 BlockCuts() [1/2]

```

template<class Cut = CuttingStrategy::Block, class Strat = StrategyParameter::Threads>
void BlockCuts (
 size_t & RBLOCKSIZE,
 size_t & CBLOCKSIZE,
 const size_t m,
 const size_t n,
 const size_t numthreads) [inline]

```

### 11.1.3.300 BlockCuts< CuttingStrategy::Single, StrategyParameter::Threads >()

```

template<>
void BlockCuts< CuttingStrategy::Single, StrategyParameter::Threads > (

```

```

 size_t & RBLOCKSIZE,
 size_t & CBLOCKSIZE,
 const size_t m,
 const size_t n,
 const size_t numthreads) [inline]

```

#### 11.1.3.301 BlockCuts< CuttingStrategy::Row, StrategyParameter::Fixed >()

```

template<>
void BlockCuts< CuttingStrategy::Row, StrategyParameter::Fixed > (
 size_t & RBLOCKSIZE,
 size_t & CBLOCKSIZE,
 const size_t m,
 const size_t n,
 const size_t numthreads) [inline]

```

#### 11.1.3.302 BlockCuts< CuttingStrategy::Row, StrategyParameter::Grain >()

```

template<>
void BlockCuts< CuttingStrategy::Row, StrategyParameter::Grain > (
 size_t & RBLOCKSIZE,
 size_t & CBLOCKSIZE,
 const size_t m,
 const size_t n,
 const size_t grainsize) [inline]

```

#### 11.1.3.303 BlockCuts< CuttingStrategy::Block, StrategyParameter::Grain >()

```

template<>
void BlockCuts< CuttingStrategy::Block, StrategyParameter::Grain > (
 size_t & RBLOCKSIZE,
 size_t & CBLOCKSIZE,
 const size_t m,
 const size_t n,
 const size_t grainsize) [inline]

```

#### 11.1.3.304 BlockCuts< CuttingStrategy::Column, StrategyParameter::Fixed >()

```

template<>
void BlockCuts< CuttingStrategy::Column, StrategyParameter::Fixed > (
 size_t & RBLOCKSIZE,
 size_t & CBLOCKSIZE,
 const size_t m,
 const size_t n,
 const size_t numthreads) [inline]

```

**11.1.3.305 BlockCuts< CuttingStrategy::Column, StrategyParameter::Grain >()**

```
template<>
void BlockCuts< CuttingStrategy::Column, StrategyParameter::Grain > (
 size_t & RBLOCKSIZE,
 size_t & CBLOCKSIZE,
 const size_t m,
 const size_t n,
 const size_t grainsize) [inline]
```

**11.1.3.306 BlockCuts< CuttingStrategy::Block, StrategyParameter::Fixed >()**

```
template<>
void BlockCuts< CuttingStrategy::Block, StrategyParameter::Fixed > (
 size_t & RBLOCKSIZE,
 size_t & CBLOCKSIZE,
 const size_t m,
 const size_t n,
 const size_t numthreads) [inline]
```

**11.1.3.307 BlockCuts< CuttingStrategy::Row, StrategyParameter::Threads >()**

```
template<>
void BlockCuts< CuttingStrategy::Row, StrategyParameter::Threads > (
 size_t & RBLOCKSIZE,
 size_t & CBLOCKSIZE,
 const size_t m,
 const size_t n,
 const size_t numthreads) [inline]
```

**11.1.3.308 BlockCuts< CuttingStrategy::Column, StrategyParameter::Threads >()**

```
template<>
void BlockCuts< CuttingStrategy::Column, StrategyParameter::Threads > (
 size_t & RBLOCKSIZE,
 size_t & CBLOCKSIZE,
 const size_t m,
 const size_t n,
 const size_t numthreads) [inline]
```

**11.1.3.309 BlockCuts< CuttingStrategy::Block, StrategyParameter::Threads >()**

```
template<>
void BlockCuts< CuttingStrategy::Block, StrategyParameter::Threads > (
 size_t & RBLOCKSIZE,
 size_t & CBLOCKSIZE,
 const size_t m,
 const size_t n,
 const size_t numthreads) [inline]
```

**11.1.3.310 BlockCuts()** [2/2]

```

template<class Cut = CuttingStrategy::Block, class Param = StrategyParameter::Threads>
void BlockCuts (
 size_t & rowBlockSize,
 size_t & colBlockSize,
 size_t & lastRBS,
 size_t & lastCBS,
 size_t & changeRBS,
 size_t & changeCBS,
 size_t & numRowsBlock,
 size_t & numColBlock,
 size_t m,
 size_t n,
 const size_t numthreads) [inline]

```

**11.1.3.311 pfzero()**

```

template<class Field >
void pfzero (
 const Field & F,
 size_t m,
 size_t n,
 typename Field::Element_ptr C,
 size_t BS = 0)

```

**11.1.3.312 pfrand()**

```

template<class Field , class RandIter >
void pfrand (
 const Field & F,
 RandIter & G,
 size_t m,
 size_t n,
 typename Field::Element_ptr C,
 size_t BS = 0)

```

**11.1.3.313 fdot()** [11/11]

```

template<class Field , class Cut , class Param >
Field::Element & fdot (
 const Field & F,
 const size_t N,
 typename Field::ConstElement_ptr x,
 const size_t incx,
 typename Field::ConstElement_ptr y,
 const size_t incy,
 typename Field::Element & d,
 const ParSeqHelper::Parallel< Cut, Param > par) [inline]

```

**11.1.3.314 pfgemm() [2/7]**

```

template<class Field , class AlgoT , class FieldTrait >
Field::Element * pfgemm (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const typename Field::Element alpha,
 const typename Field::ConstElement_ptr A,
 const size_t lda,
 const typename Field::ConstElement_ptr B,
 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element * C,
 const size_t ldc,
 MMHelper< Field, AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Block,
StrategyParameter::Threads > > & H)

```

**11.1.3.315 pfgemm() [3/7]**

```

template<class Field , class AlgoT , class FieldTrait >
Field::Element * pfgemm (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const typename Field::Element alpha,
 const typename Field::ConstElement_ptr AA,
 const size_t lda,
 const typename Field::ConstElement_ptr BB,
 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element * C,
 const size_t ldc,
 MMHelper< Field, AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Recursive,
StrategyParameter::ThreeDAdaptive > > & H)

```

**11.1.3.316 pfgemm() [4/7]**

```

template<class Field , class AlgoT , class FieldTrait >
Field::Element * pfgemm (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const typename Field::Element alpha,
 const typename Field::ConstElement_ptr AA,

```

```

 const size_t lda,
 const typename Field::ConstElement_ptr BB,
 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element * C,
 const size_t ldc,
 MMHelper< Field, AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Recursive,
StrategyParameter::TwoDAdaptive > > & H)

```

#### 11.1.3.317 pfgemm() [5/7]

```

template<class Field , class AlgoT , class FieldTrait >
Field::Element * pfgemm (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const typename Field::Element alpha,
 const typename Field::ConstElement_ptr AA,
 const size_t lda,
 const typename Field::ConstElement_ptr BB,
 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element * C,
 const size_t ldc,
 MMHelper< Field, AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Recursive,
StrategyParameter::TwoD > > & H)

```

#### 11.1.3.318 pfgemm() [6/7]

```

template<class Field , class AlgoT , class FieldTrait >
Field::Element_ptr pfgemm (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const typename Field::Element alpha,
 const typename Field::ConstElement_ptr A,
 const size_t lda,
 const typename Field::ConstElement_ptr B,
 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 MMHelper< Field, AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Recursive,
StrategyParameter::ThreeD > > & H)

```



**11.1.3.319 pfgemm()** [7/7]

```

template<class Field , class AlgoT , class FieldTrait >
Field::Element * pfgemm (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const typename Field::Element alpha,
 const typename Field::ConstElement_ptr A,
 const size_t lda,
 const typename Field::ConstElement_ptr B,
 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 MMHelper< Field, AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Recursive,
StrategyParameter::ThreeDInPlace > > & H)

```

**11.1.3.320 fgemv()** [18/19]

```

template<class Field , class AlgoT , class FieldTrait >
Field::Element_ptr fgemv (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const size_t m,
 const size_t n,
 const typename Field::Element alpha,
 const typename Field::ConstElement_ptr A,
 const size_t lda,
 const typename Field::ConstElement_ptr X,
 const size_t incX,
 const typename Field::Element beta,
 typename Field::Element_ptr Y,
 const size_t incY,
 MMHelper< Field, AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Recursive,
StrategyParameter::Threads > > & H)

```

**11.1.3.321 fgemv()** [19/19]

```

template<class Field , class AlgoT , class FieldTrait , class Cut >
Field::Element_ptr fgemv (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const size_t m,
 const size_t n,
 const typename Field::Element alpha,
 const typename Field::ConstElement_ptr A,
 const size_t lda,
 const typename Field::ConstElement_ptr X,
 const size_t incX,
 const typename Field::Element beta,

```

```

 typename Field::Element_ptr Y,
 const size_t incY,
 MMHelper< Field, AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Row,
Cut > > & H)

```

#### 11.1.3.322 parseArguments()

```

void parseArguments (
 int argc,
 char ** argv,
 Argument * args,
 bool printDefaults = true)

```

#### 11.1.3.323 getArgumentValue()

```

char * getArgumentValue (
 int argc,
 char ** argv,
 int i)

```

Get the value of an argument and avoid core dump when no value was given after an argument.

##### Parameters

|             |                     |
|-------------|---------------------|
| <i>argv</i> | argument value list |
| <i>i</i>    | argument index      |

##### Returns

char\* argument value

#### 11.1.3.324 writeCommandString()

```

std::ostream & writeCommandString (
 std::ostream & os,
 Argument * args,
 const char * programName = nullptr)

```

writes the values of all arguments, preceded by the programName

#### 11.1.3.325 WriteMatrix() [1/2]

```

template<class Field >
std::ostream & WriteMatrix (
 std::ostream & c,
 const Field & F,
 size_t m,
 size_t n,
 typename Field::ConstElement_ptr A,

```

```
size_t lda,
FFLAS_FORMAT format,
bool column_major) [inline]
```

WriteMatrix: write a matrix to an output stream.

## Parameters

|                     |                                                                      |
|---------------------|----------------------------------------------------------------------|
| <i>c</i>            | output stream                                                        |
| <i>F</i>            | base field                                                           |
| <i>m</i>            | row dimension                                                        |
| <i>n</i>            | column dimension                                                     |
| <i>A</i>            | matrix                                                               |
| <i>format</i>       | input format (FflasAuto, FflasDense, FflasSMS, FflasBinary)          |
| <i>column_major</i> | whether the matrix is stored in column or row major (row by default) |

**11.1.3.326 preamble()**

```
void preamble (
 std::ifstream & ifs,
 FFLAS_FORMAT & format) [inline]
```

**11.1.3.327 ReadMatrix()** [1/2]

```
template<class Field >
Field::Element_ptr ReadMatrix (
 std::ifstream & ifs,
 Field & F,
 size_t & m,
 size_t & n,
 typename Field::Element_ptr & A,
 FFLAS_FORMAT format = FflasAuto)
```

ReadMatrix: read a matrix from an input stream.

## Parameters

|     |               |                                                             |
|-----|---------------|-------------------------------------------------------------|
|     | <i>ifs</i>    | input stream                                                |
|     | <i>F</i>      | base field                                                  |
| out | <i>m</i>      | row dimension                                               |
| out | <i>n</i>      | column dimension                                            |
| out | <i>A</i>      | output matrix                                               |
|     | <i>format</i> | input format (FflasAuto, FflasDense, FflasSMS, FflasBinary) |

**11.1.3.328 ReadMatrix()** [2/2]

```
template<class Field >
Field::Element_ptr ReadMatrix (
 const std::string & matrix_file,
 Field & F,
 size_t & m,
 size_t & n,
 typename Field::Element_ptr & A,
 FFLAS_FORMAT format = FflasAuto) [inline]
```

ReadMatrix: read a matrix from a file.

## Parameters

|     |                    |                                                             |
|-----|--------------------|-------------------------------------------------------------|
|     | <i>matrix_file</i> | filename                                                    |
|     | <i>F</i>           | base field                                                  |
| out | <i>m</i>           | row dimension                                               |
| out | <i>n</i>           | column dimension                                            |
| out | <i>A</i>           | output matrix                                               |
|     | <i>format</i>      | input format (FflasAuto, FflasDense, FflasSMS, FflasBinary) |

## 11.1.3.329 WriteMatrix() [2/2]

```
template<class Field >
void WriteMatrix (
 std::string & matrix_file,
 const Field & F,
 int m,
 int n,
 typename Field::ConstElement_ptr A,
 size_t lda,
 FFLAS_FORMAT format = FflasDense,
 bool column_major = false)
```

WriteMatrix: write a matrix to a file.

## Parameters

|                     |                                                                      |
|---------------------|----------------------------------------------------------------------|
| <i>matrix_file</i>  | file name                                                            |
| <i>F</i>            | base field                                                           |
| <i>m</i>            | row dimension                                                        |
| <i>n</i>            | column dimension                                                     |
| <i>A</i>            | matrix                                                               |
| <i>format</i>       | input format (FflasAuto, FflasDense, FflasSMS, FflasBinary)          |
| <i>column_major</i> | whether the matrix is stored in column or row major (row by default) |

## 11.1.3.330 WritePermutation()

```
std::ostream & WritePermutation (
 std::ostream & c,
 const size_t * P,
 size_t N) [inline]
```

WritePermutation: write a permutation matrix to an output stream.

## Parameters

|          |                         |
|----------|-------------------------|
| <i>c</i> | output stream           |
| <i>P</i> | permutation             |
| <i>N</i> | size of the permutation |

**11.1.3.331 alignable()**

```
template<class Element >
bool alignable () [inline]
```

**11.1.3.332 alignable< Givaro::Integer \* >()**

```
template<>
bool alignable< Givaro::Integer * > () [inline]
```

**11.1.3.333 fflas\_new() [5/7]**

```
template<class Field >
Field::Element_ptr fflas_new (
 const Field & F,
 const size_t m,
 const Alignment align = Alignment::DEFAULT) [inline]
```

**11.1.3.334 fflas\_new() [6/7]**

```
template<class Field >
Field::Element_ptr fflas_new (
 const Field & F,
 const size_t m,
 const size_t n,
 const Alignment align = Alignment::DEFAULT) [inline]
```

**11.1.3.335 fflas\_new() [7/7]**

```
template<class Element >
Element * fflas_new (
 const size_t m,
 const Alignment align = Alignment::DEFAULT) [inline]
```

**11.1.3.336 fflas\_delete() [3/4]**

```
template<class Element_ptr >
void fflas_delete (
 Element_ptr A) [inline]
```

**11.1.3.337 fflas\_delete() [4/4]**

```
template<class Ptr , class ... Args>
void fflas_delete (
 Ptr p,
 Args ... args) [inline]
```

**11.1.3.338 prefetch()**

```
void prefetch (
 const int64_t *) [inline]
```

**11.1.3.339 getTLBSize()**

```
void getTLBSize (
 int & tlb) [inline]
```

**11.1.3.340 queryCacheSizes()**

```
void queryCacheSizes (
 int & l1,
 int & l2,
 int & l3) [inline]
```

Queries and returns the cache sizes in Bytes of the L1, L2, and L3 data caches respectively

**11.1.3.341 queryL1CacheSize()**

```
int queryL1CacheSize () [inline]
```

**Returns**

the size in Bytes of the L1 data cache

**11.1.3.342 queryTopLevelCacheSize()**

```
int queryTopLevelCacheSize () [inline]
```

**Returns**

the size in Bytes of the L2 or L3 cache if this later is present

**11.1.3.343 getSeed()**

```
uint64_t getSeed ()
```

## 11.2 FFLAS::\_ftranspose\_impl Namespace Reference

### Functions

- `template<size_t bs, typename Field , typename BTSimd >`  
`void not_inplace (const Field &F, const BTSimd &BTS, const size_t m, const size_t n, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb)`
- `template<size_t bs, typename Field , typename BTSimd >`  
`void square_inplace (const Field &F, const BTSimd &BTS, const size_t m, typename Field::Element_ptr A, const size_t lda)`
- `template<size_t bs, typename Field , typename BTSimd >`  
`void nonsquare_inplace_v1 (const Field &F, const BTSimd &BTS, const size_t m, const size_t n, typename Field::Element_ptr A)`
- `template<size_t bs, typename Field , typename BTSimd >`  
`void nonsquare_inplace_v2 (const Field &F, const BTSimd &BTS, const size_t m, const size_t n, typename Field::Element_ptr A)`

### 11.2.1 Function Documentation

#### 11.2.1.1 not\_inplace()

```
template<size_t bs, typename Field , typename BTSimd >
void not_inplace (
 const Field & F,
 const BTSimd & BTS,
 const size_t m,
 const size_t n,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::Element_ptr B,
 const size_t ldb)
```

#### 11.2.1.2 square\_inplace()

```
template<size_t bs, typename Field , typename BTSimd >
void square_inplace (
 const Field & F,
 const BTSimd & BTS,
 const size_t m,
 typename Field::Element_ptr A,
 const size_t lda)
```

#### 11.2.1.3 nonsquare\_inplace\_v1()

```
template<size_t bs, typename Field , typename BTSimd >
void nonsquare_inplace_v1 (
 const Field & F,
 const BTSimd & BTS,
 const size_t m,
 const size_t n,
 typename Field::Element_ptr A)
```



### 11.2.1.4 nonsquare\_inplace\_v2()

```
template<size_t bs, typename Field , typename BTSimd >
void nonsquare_inplace_v2 (
 const Field & F,
 const BTSimd & BTS,
 const size_t m,
 const size_t n,
 typename Field::Element_ptr A)
```

## 11.3 FFLAS::BLAS3 Namespace Reference

### Functions

- template<class Field >
   
void Bini (const Field &F, const FFLAS\_TRANSPOSE ta, const FFLAS\_TRANSPOSE tb, const size\_t mr,
   
const size\_t nr, const size\_t kr, const typename Field::Element alpha, const typename Field::Element\_ptr A,
   
const size\_t lda, const typename Field::Element\_ptr B, const size\_t ldb, const typename Field::Element beta,
   
typename Field::Element\_ptr C, const size\_t ldc, const size\_t kmax, const size\_t w, const FFLAS\_BASE
   
base, const size\_t rec\_level)
- template<class Field , class FieldTrait , class Strat , class Param >
   
Field::Element\_ptr WinoPar (const Field &F, const FFLAS\_TRANSPOSE ta, const FFLAS\_TRANSPOSE
   
tb, const size\_t mr, const size\_t nr, const size\_t kr, const typename Field::Element alpha, typename
   
Field::ConstElement\_ptr A, const size\_t lda, typename Field::ConstElement\_ptr B, const size\_t ldb,
   
const typename Field::Element beta, typename Field::Element\_ptr C, const size\_t ldc, MMHelper< Field,
   
MMHelperAlgo::WinogradPar, FieldTrait, ParSeqHelper::Parallel< Strat, Param > > &WH)
- template<class Field , class FieldTrait >
   
void Winograd (const Field &F, const FFLAS\_TRANSPOSE ta, const FFLAS\_TRANSPOSE tb, const size\_t
   
mr, const size\_t nr, const size\_t kr, const typename Field::Element alpha, typename Field::ConstElement\_ptr
   
A, const size\_t lda, typename Field::ConstElement\_ptr B, const size\_t ldb, const typename Field::Element
   
beta, typename Field::Element\_ptr C, const size\_t ldc, MMHelper< Field, MMHelperAlgo::Winograd,
   
FieldTrait > &WH)
- template<class Field , class FieldTrait >
   
void WinogradAcc\_3\_23 (const Field &F, const FFLAS\_TRANSPOSE ta, const FFLAS\_TRANSPOSE
   
tb, const size\_t mr, const size\_t nr, const size\_t kr, const typename Field::Element alpha, typename
   
Field::ConstElement\_ptr A, const size\_t lda, typename Field::ConstElement\_ptr B, const size\_t ldb,
   
const typename Field::Element beta, typename Field::Element\_ptr C, const size\_t ldc, MMHelper< Field,
   
MMHelperAlgo::Winograd, FieldTrait > &WH)
- template<class Field , class FieldTrait >
   
void WinogradAcc\_3\_21 (const Field &F, const FFLAS\_TRANSPOSE ta, const FFLAS\_TRANSPOSE
   
tb, const size\_t mr, const size\_t nr, const size\_t kr, const typename Field::Element alpha, typename
   
Field::ConstElement\_ptr A, const size\_t lda, typename Field::ConstElement\_ptr B, const size\_t ldb,
   
const typename Field::Element beta, typename Field::Element\_ptr C, const size\_t ldc, MMHelper< Field,
   
MMHelperAlgo::Winograd, FieldTrait > &WH)
- template<class Field , class FieldTrait >
   
void WinogradAcc\_2\_24 (const Field &F, const FFLAS\_TRANSPOSE ta, const FFLAS\_TRANSPOSE tb,
   
const size\_t mr, const size\_t nr, const size\_t kr, const typename Field::Element alpha, const typename
   
Field::Element\_ptr A, const size\_t lda, const typename Field::Element\_ptr B, const size\_t ldb, const
   
typename Field::Element beta, typename Field::Element\_ptr C, const size\_t ldc, MMHelper< Field,
   
MMHelperAlgo::Winograd, FieldTrait > &WH)
- template<class Field , class FieldTrait >
   
void WinogradAcc\_2\_27 (const Field &F, const FFLAS\_TRANSPOSE ta, const FFLAS\_TRANSPOSE tb,
   
const size\_t mr, const size\_t nr, const size\_t kr, const typename Field::Element alpha, const typename
   
Field::Element\_ptr A, const size\_t lda, const typename Field::Element\_ptr B, const size\_t ldb, const
   
typename Field::Element beta, typename Field::Element\_ptr C, const size\_t ldc, MMHelper< Field,
   
MMHelperAlgo::Winograd, FieldTrait > &WH)

- `template<class Field , class FieldTrait >`  
`void WinogradAcc_LR (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t mr, const size_t nr, const size_t kr, const typename Field::Element alpha, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, const MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > &WH)`
- `template<class Field , class FieldTrait >`  
`void WinogradAcc_R_S (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t mr, const size_t nr, const size_t kr, const typename Field::Element alpha, const typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, const MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > &WH)`
- `template<class Field , class FieldTrait >`  
`void WinogradAcc_L_S (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t mr, const size_t nr, const size_t kr, const typename Field::Element alpha, typename Field::Element_ptr A, const size_t lda, const typename Field::Element_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, const MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > &WH)`
- `template<class Field , class FieldTrait >`  
`void Winograd_LR_S (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t mr, const size_t nr, const size_t kr, const typename Field::Element alpha, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, const MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > &WH)`
- `template<class Field , class FieldTrait >`  
`void Winograd_L_S (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t mr, const size_t nr, const size_t kr, const typename Field::Element alpha, typename Field::Element_ptr A, const size_t lda, const typename Field::Element_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, const MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > &WH)`
- `template<class Field , class FieldTrait >`  
`void Winograd_R_S (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t mr, const size_t nr, const size_t kr, const typename Field::Element alpha, const typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, const MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > &WH)`

### 11.3.1 Function Documentation

#### 11.3.1.1 Bini()

```
template<class Field >
void Bini (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t mr,
 const size_t nr,
 const size_t kr,
 const typename Field::Element alpha,
 const typename Field::Element_ptr A,
 const size_t lda,
 const typename Field::Element_ptr B,
 const size_t ldb,
 const typename Field::Element beta,
```

```

typename Field::Element_ptr C,
const size_t ldc,
const size_t kmax,
const size_t w,
const FFLAS_BASE base,
const size_t rec_level) [inline]

```

### 11.3.1.2 WinoPar()

```

template<class Field , class FieldTrait , class Strat , class Param >
Field::Element_ptr WinoPar (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t mr,
 const size_t nr,
 const size_t kr,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr B,
 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 MMHelper< Field, MMHelperAlgo::WinogradPar, FieldTrait, ParSeqHelper::Parallel<
Strat, Param > > & WH) [inline]

```

### 11.3.1.3 Winograd()

```

template<class Field , class FieldTrait >
void Winograd (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t mr,
 const size_t nr,
 const size_t kr,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr B,
 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > & WH) [inline]

```

### 11.3.1.4 WinogradAcc\_3\_23()

```

template<class Field , class FieldTrait >
void WinogradAcc_3_23 (
 const Field & F,

```

```

const FFLAS_TRANSPOSE ta,
const FFLAS_TRANSPOSE tb,
const size_t mr,
const size_t nr,
const size_t kr,
const typename Field::Element alpha,
typename Field::ConstElement_ptr A,
const size_t lda,
typename Field::ConstElement_ptr B,
const size_t ldb,
const typename Field::Element beta,
typename Field::Element_ptr C,
const size_t ldc,
MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > & WH) [inline]

```

#### 11.3.1.5 WinogradAcc\_3\_21()

```

template<class Field , class FieldTrait >
void WinogradAcc_3_21 (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t mr,
 const size_t nr,
 const size_t kr,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr B,
 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > & WH) [inline]

```

#### 11.3.1.6 WinogradAcc\_2\_24()

```

template<class Field , class FieldTrait >
void WinogradAcc_2_24 (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t mr,
 const size_t nr,
 const size_t kr,
 const typename Field::Element alpha,
 const typename Field::Element_ptr A,
 const size_t lda,
 const typename Field::Element_ptr B,
 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > & WH) [inline]

```

### 11.3.1.7 WinogradAcc\_2\_27()

```

template<class Field , class FieldTrait >
void WinogradAcc_2_27 (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t mr,
 const size_t nr,
 const size_t kr,
 const typename Field::Element alpha,
 const typename Field::Element_ptr A,
 const size_t lda,
 const typename Field::Element_ptr B,
 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > & WH) [inline]

```

### 11.3.1.8 WinogradAcc\_LR()

```

template<class Field , class FieldTrait >
void WinogradAcc_LR (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t mr,
 const size_t nr,
 const size_t kr,
 const typename Field::Element alpha,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr B,
 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 const MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > & WH) [inline]

```

### 11.3.1.9 WinogradAcc\_R\_S()

```

template<class Field , class FieldTrait >
void WinogradAcc_R_S (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t mr,
 const size_t nr,
 const size_t kr,
 const typename Field::Element alpha,
 const typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr B,

```

```

 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 const MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > & WH) [inline]

```

#### 11.3.1.10 WinogradAcc\_L\_S()

```

template<class Field , class FieldTrait >
void WinogradAcc_L_S (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t mr,
 const size_t nr,
 const size_t kr,
 const typename Field::Element alpha,
 typename Field::Element_ptr A,
 const size_t lda,
 const typename Field::Element_ptr B,
 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 const MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > & WH) [inline]

```

#### 11.3.1.11 Winograd\_LR\_S()

```

template<class Field , class FieldTrait >
void Winograd_LR_S (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t mr,
 const size_t nr,
 const size_t kr,
 const typename Field::Element alpha,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr B,
 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 const MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > & WH) [inline]

```

#### 11.3.1.12 Winograd\_L\_S()

```

template<class Field , class FieldTrait >
void Winograd_L_S (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,

```

```

const size_t mr,
const size_t nr,
const size_t kr,
const typename Field::Element alpha,
typename Field::Element_ptr A,
const size_t lda,
const typename Field::Element_ptr B,
const size_t ldb,
const typename Field::Element beta,
typename Field::Element_ptr C,
const size_t ldc,
const MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > & WH) [inline]

```

### 11.3.1.13 Winograd\_R\_S()

```

template<class Field , class FieldTrait >
void Winograd_R_S (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t mr,
 const size_t nr,
 const size_t kr,
 const typename Field::Element alpha,
 const typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr B,
 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 const MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > & WH) [inline]

```

## 11.4 FFLAS::csr\_hyb\_details Namespace Reference

### Data Structures

- struct [Coo](#)
- struct [Info](#)

## 11.5 FFLAS::CuttingStrategy Namespace Reference

### Data Structures

- struct [Block](#)
- struct [Column](#)
- struct [Recursive](#)
- struct [Row](#)
- struct [Single](#)

## Typedefs

- [typedef Row RNSModulus](#)

## 11.5.1 Typedef Documentation

### 11.5.1.1 RNSModulus

```
typedef Row RNSModulus
```

## 11.6 FFLAS::details Namespace Reference

### Functions

- `template<class Field , bool ADD>`  
`std::enable_if< FFLAS::support_simd_add< typenameField::Element >::value, void >::type fadd`  
`(const Field &F, const size_t N, typename Field::ConstElement_ptr A, const size_t inca, typename`  
`Field::ConstElement_ptr B, const size_t incb, typename Field::Element_ptr C, const size_t incc,`  
`FieldCategories::ModularTag)`
- `template<class Field , bool ADD>`  
`std::enable_if<!FFLAS::support_simd_add< typenameField::Element >::value, void >::type fadd`  
`(const Field &F, const size_t N, typename Field::ConstElement_ptr A, const size_t inca, typename`  
`Field::ConstElement_ptr B, const size_t incb, typename Field::Element_ptr C, const size_t incc,`  
`FieldCategories::ModularTag)`
- `template<class Field , bool ADD>`  
`void fadd (const Field &F, const size_t N, typename Field::ConstElement_ptr A, const size_t inca,`  
`typename Field::ConstElement_ptr B, const size_t incb, typename Field::Element_ptr C, const size_t incc,`  
`FieldCategories::GenericTag)`
- `template<class Field , bool ADD>`  
`std::enable_if<!FFLAS::support_simd_add< typenameField::Element >::value, void >::type fadd`  
`(const Field &F, const size_t N, typename Field::ConstElement_ptr A, const size_t inca, typename`  
`Field::ConstElement_ptr B, const size_t incb, typename Field::Element_ptr C, const size_t incc,`  
`FieldCategories::UnparametricTag)`
- `template<class Field , bool ADD>`  
`std::enable_if< FFLAS::support_simd_add< typenameField::Element >::value, void >::type fadd`  
`(const Field &F, const size_t N, typename Field::ConstElement_ptr A, const size_t inca, typename`  
`Field::ConstElement_ptr B, const size_t incb, typename Field::Element_ptr C, const size_t incc,`  
`FieldCategories::UnparametricTag)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type faxpy (const`  
`Field &F, const size_t N, const typename Field::Element a, typename Field::ConstElement_ptr X, const size_t`  
`incX, typename Field::Element_ptr Y, const size_t incY, FieldCategories::ModularTag)`
- `template<class Field , class FC >`  
`void faxpy (const Field &F, const size_t N, const typename Field::Element a, typename Field::ConstElement_ptr`  
`X, const size_t incX, typename Field::Element_ptr Y, const size_t incY, FC)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type freduce (const`  
`Field &F, const size_t m, typename Field::Element_ptr A, const size_t incX, FieldCategories::ModularTag)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type freduce`  
`(const Field &F, const size_t m, typename Field::ConstElement_ptr B, const size_t incY, typename`  
`Field::Element_ptr A, const size_t incX, FieldCategories::ModularTag)`



- `template<class Field , class FC >`  
`void freduce (const Field &F, const size_t m, typename Field::Element_ptr A, const size_t incX, FC)`
- `template<class Field , class FC >`  
`void freduce (const Field &F, const size_t m, typename Field::ConstElement_ptr B, const size_t incY, typename Field::Element_ptr A, const size_t incX, FC)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type fscalin (const Field &F, const size_t N, const typename Field::Element a, typename Field::Element_ptr X, const size_t incX, FieldCategories::ModularTag)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type fscal (const Field &F, const size_t N, const typename Field::Element a, typename Field::ConstElement_ptr X, const size_t incX, typename Field::Element_ptr Y, const size_t incY, FieldCategories::ModularTag)`
- `template<class Field , class FC >`  
`void fscalin (const Field &F, const size_t n, const typename Field::Element a, typename Field::Element_ptr X, const size_t incX, FC)`
- `template<class Field , class FC >`  
`void fscal (const Field &F, const size_t N, const typename Field::Element a, typename Field::ConstElement_ptr X, const size_t incX, typename Field::Element_ptr Y, const size_t incY, FC)`
- `template<enum number_kind K>`  
`void igebb44 (size_t i, size_t j, size_t depth, size_t pdeth, const int64_t alpha, const int64_t *bIA, const int64_t *bIB, int64_t *C, size_t ldc)`
- `template<enum number_kind K>`  
`void igebb24 (size_t i, size_t j, size_t depth, size_t pdeth, const int64_t alpha, const int64_t *bIA, const int64_t *bIB, int64_t *C, size_t ldc)`
- `template<enum number_kind K>`  
`void igebb14 (size_t i, size_t j, size_t depth, size_t pdeth, const int64_t alpha, const int64_t *bIA, const int64_t *bIB, int64_t *C, size_t ldc)`
- `template<enum number_kind K>`  
`void igebb41 (size_t i, size_t j, size_t depth, size_t pdeth, const int64_t alpha, const int64_t *bIA, const int64_t *bIB, int64_t *C, size_t ldc)`
- `template<enum number_kind K>`  
`void igebb21 (size_t i, size_t j, size_t depth, size_t pdeth, const int64_t alpha, const int64_t *bIA, const int64_t *bIB, int64_t *C, size_t ldc)`
- `template<enum number_kind K>`  
`void igebb11 (size_t i, size_t j, size_t depth, size_t pdeth, const int64_t alpha, const int64_t *bIA, const int64_t *bIB, int64_t *C, size_t ldc)`
- `template<enum number_kind K>`  
`void igebp (size_t rows, size_t cols, size_t depth, const int64_t alpha, const int64_t *blockA, size_t lda, const int64_t *blockB, size_t ldb, int64_t *C, size_t ldc)`
- `template<size_t k, bool transpose>`  
`void pack_lhs (int64_t *XX, const int64_t *X, size_t idx, size_t rows, size_t cols)`
- `template<size_t k, bool transpose>`  
`void pack_rhs (int64_t *XX, const int64_t *X, size_t idx, size_t rows, size_t cols)`
- `void gebp (size_t rows, size_t cols, size_t depth, int64_t *C, size_t ldc, const int64_t *blockA, size_t lda, const int64_t *BlockB, size_t ldb, int64_t *BlockW)`
- `void BlockingFactor (size_t &m, size_t &n, size_t &k)`

## 11.6.1 Function Documentation

### 11.6.1.1 fadd() [1/5]

```
template<class Field , bool ADD>
std::enable_if< FFLAS::support_simd_add< typenameField::Element >::value, void >::type fadd (
 const Field & F,
```

```

 const size_t N,
 typename Field::ConstElement_ptr A,
 const size_t inca,
 typename Field::ConstElement_ptr B,
 const size_t incb,
 typename Field::Element_ptr C,
 const size_t incc,
 FieldCategories::ModularTag)

```

#### 11.6.1.2 fadd() [2/5]

```

template<class Field , bool ADD>
std::enable_if<!FFLAS::support_simd_add< typenameField::Element >::value, void >::type fadd (
 const Field & F,
 const size_t N,
 typename Field::ConstElement_ptr A,
 const size_t inca,
 typename Field::ConstElement_ptr B,
 const size_t incb,
 typename Field::Element_ptr C,
 const size_t incc,
 FieldCategories::ModularTag)

```

#### 11.6.1.3 fadd() [3/5]

```

template<class Field , bool ADD>
void fadd (
 const Field & F,
 const size_t N,
 typename Field::ConstElement_ptr A,
 const size_t inca,
 typename Field::ConstElement_ptr B,
 const size_t incb,
 typename Field::Element_ptr C,
 const size_t incc,
 FieldCategories::GenericTag)

```

#### 11.6.1.4 fadd() [4/5]

```

template<class Field , bool ADD>
std::enable_if<!FFLAS::support_simd_add< typenameField::Element >::value, void >::type fadd (
 const Field & F,
 const size_t N,
 typename Field::ConstElement_ptr A,
 const size_t inca,
 typename Field::ConstElement_ptr B,
 const size_t incb,
 typename Field::Element_ptr C,
 const size_t incc,
 FieldCategories::UnparametricTag) [inline]

```

**11.6.1.5 fadd()** [5/5]

```
template<class Field , bool ADD>
std::enable_if< FFLAS::support_simd_add< typenameField::Element >::value, void >::type fadd (
 const Field & F,
 const size_t N,
 typename Field::ConstElement_ptr A,
 const size_t inca,
 typename Field::ConstElement_ptr B,
 const size_t incb,
 typename Field::Element_ptr C,
 const size_t incc,
 FieldCategories::UnparametricTag) [inline]
```

**11.6.1.6 faxpy()** [1/2]

```
template<class Field >
std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type faxpy
(
 const Field & F,
 const size_t N,
 const typename Field::Element a,
 typename Field::ConstElement_ptr X,
 const size_t incX,
 typename Field::Element_ptr Y,
 const size_t incY,
 FieldCategories::ModularTag) [inline]
```

**11.6.1.7 faxpy()** [2/2]

```
template<class Field , class FC >
void faxpy (
 const Field & F,
 const size_t N,
 const typename Field::Element a,
 typename Field::ConstElement_ptr X,
 const size_t incX,
 typename Field::Element_ptr Y,
 const size_t incY,
 FC) [inline]
```

**11.6.1.8 freduce()** [1/4]

```
template<class Field >
std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type freduce
(
 const Field & F,
 const size_t m,
 typename Field::Element_ptr A,
 const size_t incX,
 FieldCategories::ModularTag) [inline]
```

**11.6.1.9 freduce()** [2/4]

```

template<class Field >
std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type freduce
(
 const Field & F,
 const size_t m,
 typename Field::ConstElement_ptr B,
 const size_t incY,
 typename Field::Element_ptr A,
 const size_t incX,
 FieldCategories::ModularTag) [inline]

```

**11.6.1.10 freduce()** [3/4]

```

template<class Field , class FC >
void freduce (
 const Field & F,
 const size_t m,
 typename Field::Element_ptr A,
 const size_t incX,
 FC) [inline]

```

**11.6.1.11 freduce()** [4/4]

```

template<class Field , class FC >
void freduce (
 const Field & F,
 const size_t m,
 typename Field::ConstElement_ptr B,
 const size_t incY,
 typename Field::Element_ptr A,
 const size_t incX,
 FC) [inline]

```

**11.6.1.12 fscaln()** [1/2]

```

template<class Field >
std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type fscaln
(
 const Field & F,
 const size_t N,
 const typename Field::Element a,
 typename Field::Element_ptr X,
 const size_t incX,
 FieldCategories::ModularTag) [inline]

```

**11.6.1.13 fscal()** [1/2]

```
template<class Field >
std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type fscal
(
 const Field & F,
 const size_t N,
 const typename Field::Element a,
 typename Field::ConstElement_ptr X,
 const size_t incX,
 typename Field::Element_ptr Y,
 const size_t incY,
 FieldCategories::ModularTag) [inline]
```

**11.6.1.14 fscaln()** [2/2]

```
template<class Field , class FC >
void fscaln (
 const Field & F,
 const size_t n,
 const typename Field::Element a,
 typename Field::Element_ptr X,
 const size_t incX,
 FC) [inline]
```

**11.6.1.15 fscal()** [2/2]

```
template<class Field , class FC >
void fscal (
 const Field & F,
 const size_t N,
 const typename Field::Element a,
 typename Field::ConstElement_ptr X,
 const size_t incX,
 typename Field::Element_ptr Y,
 const size_t incY,
 FC) [inline]
```

**11.6.1.16 igebb44()**

```
template<enum number_kind K>
void igebb44 (
 size_t i,
 size_t j,
 size_t depth,
 size_t pdeth,
 const int64_t alpha,
 const int64_t * blA,
 const int64_t * blB,
 int64_t * C,
 size_t ldc) [inline]
```

**11.6.1.17 igebb24()**

```
template<enum number_kind K>
void igebb24 (
 size_t i,
 size_t j,
 size_t depth,
 size_t pdeth,
 const int64_t alpha,
 const int64_t * blA,
 const int64_t * blB,
 int64_t * C,
 size_t ldc) [inline]
```

**11.6.1.18 igebb14()**

```
template<enum number_kind K>
void igebb14 (
 size_t i,
 size_t j,
 size_t depth,
 size_t pdeth,
 const int64_t alpha,
 const int64_t * blA,
 const int64_t * blB,
 int64_t * C,
 size_t ldc) [inline]
```

**11.6.1.19 igebb41()**

```
template<enum number_kind K>
void igebb41 (
 size_t i,
 size_t j,
 size_t depth,
 size_t pdeth,
 const int64_t alpha,
 const int64_t * blA,
 const int64_t * blB,
 int64_t * C,
 size_t ldc) [inline]
```

bug ,B\_0 dans VEC\_MADD\_32 ?

bug ,B\_0 dans VEC\_MADD\_32 ?

**11.6.1.20 igebb21()**

```
template<enum number_kind K>
void igebb21 (
 size_t i,
 size_t j,
 size_t depth,
```

```

size_t pdeth,
const int64_t alpha,
const int64_t * blA,
const int64_t * blB,
int64_t * C,
size_t ldc) [inline]

```

#### 11.6.1.21 igebb11()

```

template<enum number_kind K>
void igebb11 (
 size_t i,
 size_t j,
 size_t depth,
 size_t pdeth,
 const int64_t alpha,
 const int64_t * blA,
 const int64_t * blB,
 int64_t * C,
 size_t ldc) [inline]

```

#### 11.6.1.22 igebp()

```

template<enum number_kind K>
void igebp (
 size_t rows,
 size_t cols,
 size_t depth,
 const int64_t alpha,
 const int64_t * blockA,
 size_t lda,
 const int64_t * blockB,
 size_t ldb,
 int64_t * C,
 size_t ldc)

```

#### 11.6.1.23 pack\_lhs()

```

template<size_t k, bool transpose>
void pack_lhs (
 int64_t * XX,
 const int64_t * X,
 size_t ldx,
 size_t rows,
 size_t cols)

```

**Bug** this is fassign

**Bug** this is fassign

**Bug** this is fassign

**Bug** this is fassign

### 11.6.1.24 pack\_rhs()

```
template<size_t k, bool transpose>
void pack_rhs (
 int64_t * XX,
 const int64_t * X,
 size_t ldc,
 size_t rows,
 size_t cols)
```

**Bug** this is fassign

**Bug** this is fassign

**Bug** this is fassign

**Bug** this is fassign

### 11.6.1.25 gebp()

```
void gebp (
 size_t rows,
 size_t cols,
 size_t depth,
 int64_t * C,
 size_t ldc,
 const int64_t * blockA,
 size_t lda,
 const int64_t * BlockB,
 size_t ldb,
 int64_t * BlockW)
```

### 11.6.1.26 BlockingFactor()

```
void BlockingFactor (
 size_t & m,
 size_t & n,
 size_t & k) [inline]
```

## 11.7 FFLAS::details\_spmv Namespace Reference

### Data Structures

- struct [Coo](#)



## 11.8 FFLAS::ElementCategories Namespace Reference

### Data Structures

- struct [ArbitraryPrecIntTag](#)  
*Arbitrary precision integers: GMP.*
- struct [FixedPrecIntTag](#)  
*Fixed precision integers above machine precision: Givaro::reclnt.*
- struct [GenericTag](#)  
*default is generic*
- struct [MachineFloatTag](#)  
*float or double*
- struct [MachineIntTag](#)  
*short, int, long, long long, and unsigned variants*
- struct [RNSElementTag](#)  
*Representation in a Residue Number System.*

## 11.9 FFLAS::FieldCategories Namespace Reference

Traits and categories will need to be placed in a proper file later.

### Data Structures

- struct [GenericTag](#)  
*generic ring.*
- struct [ModularTag](#)  
*This is a modular field like e.g. `Modular<T>` or `ModularBalanced<T>`*
- struct [UnparametricTag](#)  
*If the field uses a representation with infix operators.*

### 11.9.1 Detailed Description

Traits and categories will need to be placed in a proper file later.

## 11.10 FFLAS::MMHelperAlgo Namespace Reference

### Data Structures

- struct [Auto](#)
- struct [Bini](#)
- struct [Classic](#)
- struct [DivideAndConquer](#)
- struct [Winograd](#)
- struct [WinogradPar](#)

## 11.11 FFLAS::ModeCategories Namespace Reference

Specifies the mode of action for an algorithm w.r.t.

### Data Structures

- struct [ConvertTo](#)  
*Force conversion to appropriate element type of ElementCategory T.*
- struct [DefaultBoundedTag](#)  
*Use standard field operations, but keeps track of bounds on input and output.*
- struct [DefaultTag](#)  
*No specific mode of action: use standard field operations.*
- struct [DelayedTag](#)  
*Performs field operations with delayed mod reductions. Ensures result is reduced.*
- struct [LazyTag](#)  
*Performs field operations with delayed mod only when necessary. Result may not be reduced.*

### 11.11.1 Detailed Description

Specifies the mode of action for an algorithm w.r.t.

its field

## 11.12 FFLAS::ParSeqHelper Namespace Reference

[ParSeqHelper](#) for both fgemm and ftrsm.

### Data Structures

- struct [Compose](#)
- struct [Parallel](#)
- struct [Sequential](#)

### 11.12.1 Detailed Description

[ParSeqHelper](#) for both fgemm and ftrsm.

[ParSeqHelper](#) for both fgemm and ftrsm

## 11.13 FFLAS::Protected Namespace Reference

### Data Structures

- class [AreEqual](#)
- class [AreEqual< X, X >](#)
- class [ftrmmLeftLowerNoTransNonUnit](#)
- class [ftrmmLeftLowerNoTransUnit](#)
- class [ftrmmLeftLowerTransNonUnit](#)
- class [ftrmmLeftLowerTransUnit](#)
- class [ftrmmLeftUpperNoTransNonUnit](#)
- class [ftrmmLeftUpperNoTransUnit](#)
- class [ftrmmLeftUpperTransNonUnit](#)
- class [ftrmmLeftUpperTransUnit](#)
- class [ftrmmRightLowerNoTransNonUnit](#)
- class [ftrmmRightLowerNoTransUnit](#)
- class [ftrmmRightLowerTransNonUnit](#)
- class [ftrmmRightLowerTransUnit](#)
- class [ftrmmRightUpperNoTransNonUnit](#)
- class [ftrmmRightUpperNoTransUnit](#)
- class [ftrmmRightUpperTransNonUnit](#)
- class [ftrmmRightUpperTransUnit](#)
- class [ftrsmLeftLowerNoTransNonUnit](#)
- class [ftrsmLeftLowerNoTransUnit](#)
- class [ftrsmLeftLowerTransNonUnit](#)
- class [ftrsmLeftLowerTransUnit](#)
- class [ftrsmLeftUpperNoTransNonUnit](#)

*Computes the maximal size for delaying the modular reduction in a triangular system resolution.*

- class [ftrsmLeftUpperNoTransUnit](#)
- class [ftrsmLeftUpperTransNonUnit](#)
- class [ftrsmLeftUpperTransUnit](#)
- class [ftrsmRightLowerNoTransNonUnit](#)
- class [ftrsmRightLowerNoTransUnit](#)
- class [ftrsmRightLowerTransNonUnit](#)
- class [ftrsmRightLowerTransUnit](#)
- class [ftrsmRightUpperNoTransNonUnit](#)
- class [ftrsmRightUpperNoTransUnit](#)
- class [ftrsmRightUpperTransNonUnit](#)
- class [ftrsmRightUpperTransUnit](#)

### Functions

- [template<class Field >](#)  
[double computeFactorClassic](#) (const Field &F)
- [template<> double computeFactorClassic](#) (const Givaro::ModularBalanced< double > &F)
- [template<> double computeFactorClassic](#) (const Givaro::ModularBalanced< float > &F)
- [template<class Field >](#)  
[size\\_t DotProdBoundClassic](#) (const Field &F, const typename Field::Element &beta)
- [template<class Field >](#)  
[size\\_t TRSMBound](#) (const Field &)  
*TRSMBound.*
- [template<class Element >](#)  
[size\\_t TRSMBound](#) (const Givaro::Modular< Element > &F)

*Specialization for positive modular representation over float.*

- `template<class Element >`  
`size_t TRSMBound (const Givaro::ModularBalanced< Element > &F)`

*Specialization for balanced modular representation over double.*

- `template<class NewField , class Field , class FieldMode >`  
`Field::Element_ptr fgemm_convert (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field, MMHelperAlgo::Winograd, FieldMode > &H)`
- `template<class Field , class Element , class AlgoT , class ParSeqTrait >`  
`bool NeedPreAddReduction (Element &Outmin, Element &Outmax, Element &Op1min, Element &Op1max, Element &Op2min, Element &Op2max, MMHelper< Field, AlgoT, ModeCategories::LazyTag, ParSeqTrait > &WH)`
- `template<class Field , class Element , class AlgoT , class ModeT , class ParSeqTrait >`  
`bool NeedPreAddReduction (Element &Outmin, Element &Outmax, Element &Op1min, Element &Op1max, Element &Op2min, Element &Op2max, MMHelper< Field, AlgoT, ModeT, ParSeqTrait > &WH)`
- `template<class Field , class Element , class AlgoT , class ParSeqTrait >`  
`bool NeedPreSubReduction (Element &Outmin, Element &Outmax, Element &Op1min, Element &Op1max, Element &Op2min, Element &Op2max, MMHelper< Field, AlgoT, ModeCategories::LazyTag, ParSeqTrait > &WH)`
- `template<class Field , class Element , class AlgoT , class ModeT , class ParSeqTrait >`  
`bool NeedPreSubReduction (Element &Outmin, Element &Outmax, Element &Op1min, Element &Op1max, Element &Op2min, Element &Op2max, MMHelper< Field, AlgoT, ModeT, ParSeqTrait > &WH)`
- `template<class Field , class Element , class AlgoT , class ParSeqTrait >`  
`bool NeedDoublePreAddReduction (Element &Outmin, Element &Outmax, Element &Op1min, Element &Op1max, Element &Op2min, Element &Op2max, Element beta, MMHelper< Field, AlgoT, ModeCategories::LazyTag, ParSeqTrait > &WH)`
- `template<class Field , class Element , class AlgoT , class ModeT , class ParSeqTrait >`  
`bool NeedDoublePreAddReduction (Element &Outmin, Element &Outmax, Element &Op1min, Element &Op1max, Element &Op2min, Element &Op2max, Element beta, MMHelper< Field, AlgoT, ModeT, ParSeqTrait > &WH)`
- `template<class Field , class AlgoT , class ParSeqTrait >`  
`void ScalAndReduce (const Field &F, const size_t N, const typename Field::Element alpha, typename Field::Element_ptr X, const size_t incX, const MMHelper< Field, AlgoT, ModeCategories::LazyTag, ParSeqTrait > &H)`
- `template<class Field , class AlgoT , class ParSeqTrait >`  
`void ScalAndReduce (const Field &F, const size_t M, const size_t N, const typename Field::Element alpha, typename Field::Element_ptr A, const size_t lda, const MMHelper< Field, AlgoT, ModeCategories::LazyTag, ParSeqTrait > &H)`
- `template<class Field >`  
`Field::Element_ptr fsquareCommon (const Field &F, const FFLAS_TRANSPOSE ta, const size_t n, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc)`
- `template<class Field >`  
`int WinogradThreshold (const Field &F)`  
*Computes the number of recursive levels to perform.*
- `template<> int WinogradThreshold (const Givaro::Modular< float > &F)`
- `template<> int WinogradThreshold (const Givaro::ModularBalanced< double > &F)`
- `template<> int WinogradThreshold (const Givaro::ModularBalanced< float > &F)`
- `template<class Field >`  
`int WinogradSteps (const Field &F, const size_t &m)`  
*Computes the number of recursive levels to perform.*
- `template<class Field , class FieldMode >`  
`void DynamicPeeling (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const size_t mr, const size_t nr, const size_t`

- kr, const typename Field::Element alpha, typename Field::ConstElement\_ptr A, const size\_t lda, typename Field::ConstElement\_ptr B, const size\_t ldb, const typename Field::Element beta, typename Field::Element\_ptr C, const size\_t ldc, MMHelper< Field, MMHelperAlgo::Winograd, FieldMode > &H, const typename MMHelper< Field, MMHelperAlgo::Winograd, FieldMode >::DelayedField::Element Cmin, const typename MMHelper< Field, MMHelperAlgo::Winograd, FieldMode >::DelayedField::Element Cmax)
- `template<class Field, class FieldMode >`  
`void DynamicPeeling2 (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const size_t mr, const size_t nr, const size_t kr, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field, MMHelperAlgo::Winograd, FieldMode > &H, const typename MMHelper< Field, MMHelperAlgo::Winograd, FieldMode >::DelayedField::Element Cmin, const typename MMHelper< Field, MMHelperAlgo::Winograd, FieldMode >::DelayedField::Element Cmax)`
  - `template<class Field, class FieldMode >`  
`void WinogradCalc (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t mr, const size_t nr, const size_t kr, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field, MMHelperAlgo::Winograd, FieldMode > &H)`
  - `template<typename FloatElement, class Field >`  
`Field::Element_ptr fgemv_convert (const Field &F, const FFLAS_TRANSPOSE ta, const size_t M, const size_t N, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr X, const size_t incX, const typename Field::Element beta, typename Field::Element_ptr Y, const size_t incY)`
  - `template<class FloatElement, class Field >`  
`void fger_convert (const Field &F, const size_t M, const size_t N, const typename Field::Element alpha, typename Field::ConstElement_ptr x, const size_t incx, typename Field::ConstElement_ptr y, const size_t incy, typename Field::Element_ptr A, const size_t lda)`
  - `template<class NewField, class Field, class FieldMode >`  
`Field::Element_ptr fsyrk_convert (const Field &F, const FFLAS_UPLO UpLo, const FFLAS_TRANSPOSE trans, const size_t N, const size_t K, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field, MMHelperAlgo::Classic, FieldMode > &H)`
  - `template<class Field, class AlgoT, class ParSeqTrait >`  
`void ScalAndReduce (const Field &F, const FFLAS_UPLO UpLo, const size_t N, const typename Field::Element alpha, typename Field::Element_ptr A, const size_t lda, const MMHelper< Field, AlgoT, ModeCategories::LazyTag, ParSeqTrait > &H)`
  - `template<class Field, class Element, class AlgoT, class ParSeqTrait >`  
`bool NeedPreScalReduction (Element &Outmin, Element &Outmax, Element &Op1min, Element &Op1max, const Element &x, MMHelper< Field, AlgoT, ModeCategories::LazyTag, ParSeqTrait > &WH)`
  - `template<class Field, class Element, class AlgoT, class ModeT, class ParSeqTrait >`  
`bool NeedPreScalReduction (Element &Outmin, Element &Outmax, Element &Op1min, Element &Op1max, const Element &x, MMHelper< Field, AlgoT, ModeT, ParSeqTrait > &WH)`
  - `template<class Field, class Element, class AlgoT, class ParSeqTrait >`  
`bool NeedPreApxyReduction (Element &Outmin, Element &Outmax, Element &Op1min, Element &Op1max, Element &Op2min, Element &Op2max, const Element &x, MMHelper< Field, AlgoT, ModeCategories::LazyTag, ParSeqTrait > &WH)`
  - `template<class Field, class Element, class AlgoT, class ModeT, class ParSeqTrait >`  
`bool NeedPreApxyReduction (Element &Outmin, Element &Outmax, Element &Op1min, Element &Op1max, Element &Op2min, Element &Op2max, const Element &x, MMHelper< Field, AlgoT, ModeT, ParSeqTrait > &WH)`
  - `template<class DFE >`  
`size_t min_types (const DFE &k)`
  - `template<>` `size_t min_types (const Reclnt::rint< 6 > &k)`
  - `template<>` `size_t min_types (const Reclnt::rint< 7 > &k)`
  - `template<>` `size_t min_types (const Reclnt::rint< 8 > &k)`
  - `template<>` `size_t min_types (const Reclnt::rint< 9 > &k)`

- `template<> size_t min_types (const Reclnt::rint< 10 > &k)`
- `template<> size_t min_types (const Givaro::Integer &k)`
- `template<class T >`  
`bool unfit (T x)`
- `template<> bool unfit (int64_t x)`
- `template<size_t K>`  
`bool unfit (Reclnt::rint< K > x)`
- `template<> bool unfit (Reclnt::rint< 6 > x)`
- `template<enum FFLAS_TRANSPOSE tA, enum FFLAS_TRANSPOSE tB>`  
`void igemm_colmajor (size_t rows, size_t cols, size_t depth, const int64_t alpha, const int64_t *A, size_t lda, const int64_t *B, size_t ldb, int64_t *C, size_t ldc)`
- `template<enum FFLAS_TRANSPOSE tA, enum FFLAS_TRANSPOSE tB, enum number_kind alpha_kind>`  
`void igemm_colmajor (size_t rows, size_t cols, size_t depth, const int64_t alpha, const int64_t *A, size_t lda, const int64_t *B, size_t ldb, int64_t *C, size_t ldc)`
- `void igemm (const enum FFLAS_TRANSPOSE TransA, const enum FFLAS_TRANSPOSE TransB, size_t rows, size_t cols, size_t depth, const int64_t alpha, const int64_t *A, size_t lda, const int64_t *B, size_t ldb, const int64_t beta, int64_t *C, size_t ldc)`
- `template<class Field >`  
`void MatF2MatD_Triangular (const Field &F, Givaro::DoubleDomain::Element_ptr S, const size_t lds, typename Field::ConstElement_ptr const E, const size_t lde, const size_t m, const size_t n)`
- `template<class Field >`  
`void MatF2MatFI_Triangular (const Field &F, Givaro::FloatDomain::Element_ptr S, const size_t lds, typename Field::ConstElement_ptr const E, const size_t lde, const size_t m, const size_t n)`

## 11.13.1 Function Documentation

### 11.13.1.1 computeFactorClassic() [1/3]

```
template<class Field >
double computeFactorClassic (
 const Field & F) [inline]
```

### 11.13.1.2 computeFactorClassic() [2/3]

```
template<>
double computeFactorClassic (
 const Givaro::ModularBalanced< double > & F) [inline]
```

### 11.13.1.3 computeFactorClassic() [3/3]

```
template<>
double computeFactorClassic (
 const Givaro::ModularBalanced< float > & F) [inline]
```

### 11.13.1.4 DotProdBoundClassic()

```
template<class Field >
size_t DotProdBoundClassic (
 const Field & F,
 const typename Field::Element & beta) [inline]
```

**11.13.1.5 TRSMBound()** [1/3]

```
template<class Field >
size_t TRSMBound (
 const Field &) [inline]
```

TRSMBound.

computes the maximal size for delaying the modular reduction in a triangular system resolution

This is the default version over an arbitrary field. It is currently never used (the recursive algorithm is run until  $n=1$  in this case)

**Parameters**

|     |                                      |
|-----|--------------------------------------|
| $F$ | Finite Field/Ring of the computation |
|-----|--------------------------------------|

**11.13.1.6 TRSMBound()** [2/3]

```
template<class Element >
size_t TRSMBound (
 const Givaro::Modular< Element > & F) [inline]
```

Specialization for positive modular representation over float.

Computes  $n_{\max}$  s.t.  $(p-1)/2 * (p^{\{n_{\max}-1\}} + (p-2)^{\{n_{\max}-1\}}) < 2^{24}$  @pbi See [Dumas Giorgi Pernet 06, arXiv:cs/0601133]

**11.13.1.7 TRSMBound()** [3/3]

```
template<class Element >
size_t TRSMBound (
 const Givaro::ModularBalanced< Element > & F) [inline]
```

Specialization for balanced modular representation over double.

Computes  $n_{\max}$  s.t.  $(p-1)/2 * (((p+1)/2)^{\{n_{\max}-1\}}) < 2^{53}$

**Bibliography** • Dumas Giorgi Pernet 06, arXiv:cs/0601133

**11.13.1.8 fgemm\_convert()**

```
template<class NewField , class Field , class FieldMode >
Field::Element_ptr fgemm_convert (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr B,
 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 MMHelper< Field, MMHelperAlgo::Winograd, FieldMode > & H) [inline]
```

**11.13.1.9 NeedPreAddReduction() [1/2]**

```
template<class Field , class Element , class AlgoT , class ParSeqTrait >
bool NeedPreAddReduction (
 Element & Outmin,
 Element & Outmax,
 Element & Op1min,
 Element & Op1max,
 Element & Op2min,
 Element & Op2max,
 MMHelper< Field, AlgoT, ModeCategories::LazyTag, ParSeqTrait > & WH) [inline]
```

**11.13.1.10 NeedPreAddReduction() [2/2]**

```
template<class Field , class Element , class AlgoT , class ModeT , class ParSeqTrait >
bool NeedPreAddReduction (
 Element & Outmin,
 Element & Outmax,
 Element & Op1min,
 Element & Op1max,
 Element & Op2min,
 Element & Op2max,
 MMHelper< Field, AlgoT, ModeT, ParSeqTrait > & WH) [inline]
```

**11.13.1.11 NeedPreSubReduction() [1/2]**

```
template<class Field , class Element , class AlgoT , class ParSeqTrait >
bool NeedPreSubReduction (
 Element & Outmin,
 Element & Outmax,
 Element & Op1min,
 Element & Op1max,
 Element & Op2min,
 Element & Op2max,
 MMHelper< Field, AlgoT, ModeCategories::LazyTag, ParSeqTrait > & WH) [inline]
```



**11.13.1.12 NeedPreSubReduction()** [2/2]

```
template<class Field , class Element , class AlgoT , class ModeT , class ParSeqTrait >
bool NeedPreSubReduction (
 Element & Outmin,
 Element & Outmax,
 Element & Op1min,
 Element & Op1max,
 Element & Op2min,
 Element & Op2max,
 MMHelper< Field, AlgoT, ModeT, ParSeqTrait > & WH) [inline]
```

**11.13.1.13 NeedDoublePreAddReduction()** [1/2]

```
template<class Field , class Element , class AlgoT , class ParSeqTrait >
bool NeedDoublePreAddReduction (
 Element & Outmin,
 Element & Outmax,
 Element & Op1min,
 Element & Op1max,
 Element & Op2min,
 Element & Op2max,
 Element beta,
 MMHelper< Field, AlgoT, ModeCategories::LazyTag, ParSeqTrait > & WH) [inline]
```

**11.13.1.14 NeedDoublePreAddReduction()** [2/2]

```
template<class Field , class Element , class AlgoT , class ModeT , class ParSeqTrait >
bool NeedDoublePreAddReduction (
 Element & Outmin,
 Element & Outmax,
 Element & Op1min,
 Element & Op1max,
 Element & Op2min,
 Element & Op2max,
 Element beta,
 MMHelper< Field, AlgoT, ModeT, ParSeqTrait > & WH) [inline]
```

**11.13.1.15 ScalAndReduce()** [1/3]

```
template<class Field , class AlgoT , class ParSeqTrait >
void ScalAndReduce (
 const Field & F,
 const size_t N,
 const typename Field::Element alpha,
 typename Field::Element_ptr X,
 const size_t incX,
 const MMHelper< Field, AlgoT, ModeCategories::LazyTag, ParSeqTrait > & H) [inline]
```

**11.13.1.16 ScalAndReduce()** [2/3]

```
template<class Field , class AlgoT , class ParSeqTrait >
void ScalAndReduce (
 const Field & F,
 const size_t M,
 const size_t N,
 const typename Field::Element alpha,
 typename Field::Element_ptr A,
 const size_t lda,
 const MMHelper< Field, AlgoT, ModeCategories::LazyTag, ParSeqTrait > & H) [inline]
```

**11.13.1.17 fsquareCommon()**

```
template<class Field >
Field::Element_ptr fsquareCommon (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const size_t n,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc) [inline]
```

**11.13.1.18 WinogradThreshold()** [1/4]

```
template<class Field >
int WinogradThreshold (
 const Field & F) [inline]
```

Computes the number of recursive levels to perform.

**Parameters**

|          |                                         |
|----------|-----------------------------------------|
| <i>m</i> | the common dimension in the product AxB |
|----------|-----------------------------------------|

**11.13.1.19 WinogradThreshold()** [2/4]

```
template<>
int WinogradThreshold (
 const Givaro::Modular< float > & F) [inline]
```

**11.13.1.20 WinogradThreshold()** [3/4]

```
template<>
int WinogradThreshold (
 const Givaro::ModularBalanced< double > & F) [inline]
```

**11.13.1.21 WinogradThreshold()** [4/4]

```
template<>
int WinogradThreshold (
 const Givaro::ModularBalanced< float > & F) [inline]
```

**11.13.1.22 WinogradSteps()**

```
template<class Field >
int WinogradSteps (
 const Field & F,
 const size_t & m) [inline]
```

Computes the number of recursive levels to perform.

**Parameters**

|     |                                                  |
|-----|--------------------------------------------------|
| $m$ | the common dimension in the product $A \times B$ |
|-----|--------------------------------------------------|

**11.13.1.23 DynamicPeeling()**

```
template<class Field , class FieldMode >
void DynamicPeeling (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const size_t mr,
 const size_t nr,
 const size_t kr,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr B,
 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 MMHelper< Field, MMHelperAlgo::Winograd, FieldMode > & H,
 const typename MMHelper< Field, MMHelperAlgo::Winograd, FieldMode >::Delayed←
Field::Element Cmin,
 const typename MMHelper< Field, MMHelperAlgo::Winograd, FieldMode >::Delayed←
Field::Element Cmax) [inline]
```

**11.13.1.24 DynamicPeeling2()**

```
template<class Field , class FieldMode >
void DynamicPeeling2 (
 const Field & F,
```

```

 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const size_t mr,
 const size_t nr,
 const size_t kr,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr B,
 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 MMHelper< Field, MMHelperAlgo::Winograd, FieldMode > & H,
 const typename MMHelper< Field, MMHelperAlgo::Winograd, FieldMode >::Delayed←
Field::Element Cmin,
 const typename MMHelper< Field, MMHelperAlgo::Winograd, FieldMode >::Delayed←
Field::Element Cmax) [inline]

```

### 11.13.1.25 WinogradCalc()

```

template<class Field , class FieldMode >
void WinogradCalc (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const FFLAS_TRANSPOSE tb,
 const size_t mr,
 const size_t nr,
 const size_t kr,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr B,
 const size_t ldb,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 MMHelper< Field, MMHelperAlgo::Winograd, FieldMode > & H) [inline]

```

### 11.13.1.26 fgemv\_convert()

```

template<typename FloatElement , class Field >
Field::Element_ptr fgemv_convert (
 const Field & F,
 const FFLAS_TRANSPOSE ta,
 const size_t M,
 const size_t N,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr X,
 const size_t incX,

```

```

 const typename Field::Element beta,
 typename Field::Element_ptr Y,
 const size_t incY) [inline]

```

#### 11.13.1.27 fger\_convert()

```

template<class FloatElement , class Field >
void fger_convert (
 const Field & F,
 const size_t M,
 const size_t N,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr x,
 const size_t incx,
 typename Field::ConstElement_ptr y,
 const size_t incy,
 typename Field::Element_ptr A,
 const size_t lda) [inline]

```

#### 11.13.1.28 fsyrk\_convert()

```

template<class NewField , class Field , class FieldMode >
Field::Element_ptr fsyrk_convert (
 const Field & F,
 const FFLAS_UPLO UpLo,
 const FFLAS_TRANSPOSE trans,
 const size_t N,
 const size_t K,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc,
 MMHelper< Field, MMHelperAlgo::Classic, FieldMode > & H) [inline]

```

#### 11.13.1.29 ScalAndReduce() [3/3]

```

template<class Field , class AlgoT , class ParSeqTrait >
void ScalAndReduce (
 const Field & F,
 const FFLAS_UPLO UpLo,
 const size_t N,
 const typename Field::Element alpha,
 typename Field::Element_ptr A,
 const size_t lda,
 const MMHelper< Field, AlgoT, ModeCategories::LazyTag, ParSeqTrait > & H) [inline]

```

**11.13.1.30 NeedPreScalReduction()** [1/2]

```
template<class Field , class Element , class AlgoT , class ParSeqTrait >
bool NeedPreScalReduction (
 Element & Outmin,
 Element & Outmax,
 Element & Op1min,
 Element & Op1max,
 const Element & x,
 MMHelper< Field, AlgoT, ModeCategories::LazyTag, ParSeqTrait > & WH) [inline]
```

**11.13.1.31 NeedPreScalReduction()** [2/2]

```
template<class Field , class Element , class AlgoT , class ModeT , class ParSeqTrait >
bool NeedPreScalReduction (
 Element & Outmin,
 Element & Outmax,
 Element & Op1min,
 Element & Op1max,
 const Element & x,
 MMHelper< Field, AlgoT, ModeT, ParSeqTrait > & WH) [inline]
```

**11.13.1.32 NeedPreAxyReduction()** [1/2]

```
template<class Field , class Element , class AlgoT , class ParSeqTrait >
bool NeedPreAxyReduction (
 Element & Outmin,
 Element & Outmax,
 Element & Op1min,
 Element & Op1max,
 Element & Op2min,
 Element & Op2max,
 const Element & x,
 MMHelper< Field, AlgoT, ModeCategories::LazyTag, ParSeqTrait > & WH) [inline]
```

**11.13.1.33 NeedPreAxyReduction()** [2/2]

```
template<class Field , class Element , class AlgoT , class ModeT , class ParSeqTrait >
bool NeedPreAxyReduction (
 Element & Outmin,
 Element & Outmax,
 Element & Op1min,
 Element & Op1max,
 Element & Op2min,
 Element & Op2max,
 const Element & x,
 MMHelper< Field, AlgoT, ModeT, ParSeqTrait > & WH) [inline]
```

**11.13.1.34 min\_types()** [1/7]

```
template<class DFE >
size_t min_types (
 const DFE & k) [inline]
```

**11.13.1.35 min\_types()** [2/7]

```
template<>
size_t min_types (
 const RecInt::rint< 6 > & k) [inline]
```

**11.13.1.36 min\_types()** [3/7]

```
template<>
size_t min_types (
 const RecInt::rint< 7 > & k) [inline]
```

**11.13.1.37 min\_types()** [4/7]

```
template<>
size_t min_types (
 const RecInt::rint< 8 > & k) [inline]
```

**11.13.1.38 min\_types()** [5/7]

```
template<>
size_t min_types (
 const RecInt::rint< 9 > & k) [inline]
```

**11.13.1.39 min\_types()** [6/7]

```
template<>
size_t min_types (
 const RecInt::rint< 10 > & k) [inline]
```

**11.13.1.40 min\_types()** [7/7]

```
template<>
size_t min_types (
 const Givaro::Integer & k) [inline]
```

**11.13.1.41 unfit()** [1/4]

```
template<class T >
bool unfit (
 T x) [inline]
```

**11.13.1.42 unfit()** [2/4]

```
template<>
bool unfit (
 int64_t x) [inline]
```

**11.13.1.43 unfit()** [3/4]

```
template<size_t K>
bool unfit (
 RecInt::rint< K > x) [inline]
```

**11.13.1.44 unfit()** [4/4]

```
template<>
bool unfit (
 RecInt::rint< 6 > x) [inline]
```

**11.13.1.45 igemm\_colmajor()** [1/2]

```
template<enum FFLAS_TRANSPOSE tA, enum FFLAS_TRANSPOSE tB>
void igemm_colmajor (
 size_t rows,
 size_t cols,
 size_t depth,
 const int64_t alpha,
 const int64_t * A,
 size_t lda,
 const int64_t * B,
 size_t ldb,
 int64_t * C,
 size_t ldc)
```

**11.13.1.46 igemm\_colmajor()** [2/2]

```
template<enum FFLAS_TRANSPOSE tA, enum FFLAS_TRANSPOSE tB, enum number_kind alpha_kind>
void igemm_colmajor (
 size_t rows,
 size_t cols,
 size_t depth,
 const int64_t alpha,
 const int64_t * A,
 size_t lda,
 const int64_t * B,
 size_t ldb,
 int64_t * C,
 size_t ldc)
```

**11.13.1.47 igemm()**

```
void igemm (
 const enum FFLAS_TRANSPOSE TransA,
 const enum FFLAS_TRANSPOSE TransB,
 size_t rows,
 size_t cols,
 size_t depth,
 const int64_t alpha,
```



```

const int64_t * A,
size_t lda,
const int64_t * B,
size_t ldb,
const int64_t beta,
int64_t * C,
size_t ldc) [inline]

```

**Todo** use primitive (no [Field\(\)](#)) and specialise for int64.

**Todo** use primitive (no [Field\(\)](#)) and specialise for int64.

#### 11.13.148 MatF2MatD\_Triangular()

```

template<class Field >
void MatF2MatD_Triangular (
 const Field & F,
 Givaro::DoubleDomain::Element_ptr S,
 const size_t lds,
 typename Field::ConstElement_ptr const E,
 const size_t lde,
 const size_t m,
 const size_t n)

```

#### 11.13.149 MatF2MatF1\_Triangular()

```

template<class Field >
void MatF2MatF1_Triangular (
 const Field & F,
 Givaro::FloatDomain::Element_ptr S,
 const size_t lds,
 typename Field::ConstElement_ptr const E,
 const size_t lde,
 const size_t m,
 const size_t n)

```

**Todo** do finit(...,FFLAS\_TRANS,FFLAS\_DIAG)  
do fconvert(...,FFLAS\_TRANS,FFLAS\_DIAG)

## 11.14 FFLAS::sell\_details Namespace Reference

### Data Structures

- struct [Coo](#)
- struct [Info](#)

## 11.15 FFLAS::sparse\_details Namespace Reference

### Functions

- `template<class Field >`  
`void init_y (const Field &F, const size_t m, const typename Field::Element b, typename Field::Element_ptr y)`
- `template<class Field >`  
`void init_y (const Field &F, const size_t m, const size_t n, const typename Field::Element b, typename Field::Element_ptr y, const int ldy)`
- `template<class Field , class SM , class FC , class MZO >`  
`std::enable_if<!(std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineFloat >::value)||std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineIntTag >::value)>::type fspmv_dispatch (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FC fc, MZO mzo)`
- `template<class Field , class SM , class FC , class MZO >`  
`std::enable_if< std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineFloat >::value)||std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineIntTag >::value >::type fspmv_dispatch (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FC fc, MZO mzo)`
- `template<class Field , class SM >`  
`void fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::GenericTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if<!isSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< isSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if<!isSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::ModularTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< isSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::ModularTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`void fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::GenericTag, ZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if<!isSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag, ZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< isSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag, ZOSparseMatrix)`
- `template<class Field , class SM >`  
`void fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::ModularTag, std::true_type)`
- `template<class Field , class SM , class FCat , class MZO >`  
`std::enable_if<!(std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineFloat >::value)||std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineIntTag >::value)>::type fspmm_dispatch (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FCat, MZO)`

- `template<class Field , class SM , class FCat , class MZO >`  
`std::enable_if< std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineFloat >::value || std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineIntTag >::value >::type fspmm_dispatch (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FCat, MZO)`
- `template<class Field , class SM >`  
`void fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::GenericTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< support_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::UnparametricTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< !support_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::UnparametricTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< support_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::ModularTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< !support_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::ModularTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`void fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::GenericTag, ZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< support_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::UnparametricTag, ZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< !support_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::UnparametricTag, ZOSparseMatrix)`
- `template<class Field , class SM >`  
`void fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::ModularTag, ZOSparseMatrix)`
- `template<class Field , class SM , class FCat , class MZO >`  
`std::enable_if< !(std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineFloat >::value || std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineIntTag >::value) >::type pfspmm_dispatch (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FCat, MZO)`
- `template<class Field , class SM , class FCat , class MZO >`  
`std::enable_if< std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineFloat >::value || std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineIntTag >::value >::type pfspmm_dispatch (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FCat, MZO)`
- `template<class Field , class SM >`  
`void pfspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::GenericTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< support_simd< typenameField::Element >::value >::type pfspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::UnparametricTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< !support_simd< typenameField::Element >::value >::type pfspmm (const Field &F, const SM`

- &A, size\_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::UnparametricTag, NotZOSparseMatrix)
- `template<class Field, class SM>`  
`std::enable_if< support_simd< typenameField::Element >::value >::type pfspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::ModularTag, NotZOSparseMatrix)`
  - `template<class Field, class SM>`  
`std::enable_if< !support_simd< typenameField::Element >::value >::type pfspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::ModularTag, NotZOSparseMatrix)`
  - `template<class Field, class SM>`  
`void pfspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::GenericTag, ZOSparseMatrix)`
  - `template<class Field, class SM>`  
`std::enable_if< support_simd< typenameField::Element >::value >::type pfspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::UnparametricTag, ZOSparseMatrix)`
  - `template<class Field, class SM>`  
`std::enable_if< !support_simd< typenameField::Element >::value >::type pfspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::UnparametricTag, ZOSparseMatrix)`
  - `template<class Field, class SM>`  
`void pfspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::ModularTag, ZOSparseMatrix)`
  - `template<class Field, class SM>`  
`void pfspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::GenericTag, std::false_type)`
  - `template<class Field, class SM>`  
`void pfspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag, std::false_type)`
  - `template<class Field, class SM>`  
`void pfspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::ModularTag, std::false_type)`
  - `template<class Field, class SM>`  
`void pfspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::GenericTag, std::true_type)`
  - `template<class Field, class SM>`  
`void pfspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag, std::true_type)`
  - `template<class Field, class SM>`  
`void pfspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::ModularTag, std::true_type)`
  - `template<class Field, class SM>`  
`std::enable_if< isSparseMatrixSimdFormat< Field, SM >::value &&support_simd< typenameField::Element >::value >::type fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag, NotZOSparseMatrix)`
  - `template<class Field, class SM>`  
`std::enable_if< isSparseMatrixSimdFormat< Field, SM >::value &&support_simd< typenameField::Element >::value >::type fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::ModularTag, NotZOSparseMatrix)`
  - `template<class Field, class SM>`  
`std::enable_if< isSparseMatrixSimdFormat< Field, SM >::value &&support_simd< typenameField::Element >::value >::type fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag, ZOSparseMatrix)`

## 11.15.1 Function Documentation

### 11.15.1.1 init\_y() [1/2]

```
template<class Field >
void init_y (
 const Field & F,
 const size_t m,
 const typename Field::Element b,
 typename Field::Element_ptr y) [inline]
```

### 11.15.1.2 init\_y() [2/2]

```
template<class Field >
void init_y (
 const Field & F,
 const size_t m,
 const size_t n,
 const typename Field::Element b,
 typename Field::Element_ptr y,
 const int ldy) [inline]
```

### 11.15.1.3 fspmv\_dispatch() [1/2]

```
template<class Field , class SM , class FC , class MZO >
std::enable_if<!(std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::Mac
>::value||std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineInt
>::value)>::type fspmv_dispatch (
 const Field & F,
 const SM & A,
 typename Field::ConstElement_ptr x,
 typename Field::Element_ptr y,
 FC fc,
 MZO mzo) [inline]
```

### 11.15.1.4 fspmv\_dispatch() [2/2]

```
template<class Field , class SM , class FC , class MZO >
std::enable_if< std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::Mac
>::value||std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineInt
>::value >::type fspmv_dispatch (
 const Field & F,
 const SM & A,
 typename Field::ConstElement_ptr x,
 typename Field::Element_ptr y,
 FC fc,
 MZO mzo) [inline]
```

**11.15.1.5 fspmv()** [1/12]

```
template<class Field , class SM >
void fspmv (
 const Field & F,
 const SM & A,
 typename Field::ConstElement_ptr x,
 typename Field::Element_ptr y,
 FieldCategories::GenericTag ,
 NotZOSparseMatrix) [inline]
```

**11.15.1.6 fspmv()** [2/12]

```
template<class Field , class SM >
std::enable_if<!isSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (
 const Field & F,
 const SM & A,
 typename Field::ConstElement_ptr x,
 typename Field::Element_ptr y,
 FieldCategories::UnparametricTag ,
 NotZOSparseMatrix) [inline]
```

**11.15.1.7 fspmv()** [3/12]

```
template<class Field , class SM >
std::enable_if< isSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (
 const Field & F,
 const SM & A,
 typename Field::ConstElement_ptr x,
 typename Field::Element_ptr y,
 FieldCategories::UnparametricTag ,
 NotZOSparseMatrix) [inline]
```

**11.15.1.8 fspmv()** [4/12]

```
template<class Field , class SM >
std::enable_if<!isSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (
 const Field & F,
 const SM & A,
 typename Field::ConstElement_ptr x,
 typename Field::Element_ptr y,
 FieldCategories::ModularTag ,
 NotZOSparseMatrix) [inline]
```

**11.15.1.9 fspmv()** [5/12]

```
template<class Field , class SM >
std::enable_if< isSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (
 const Field & F,
 const SM & A,
 typename Field::ConstElement_ptr x,
 typename Field::Element_ptr y,
 FieldCategories::ModularTag ,
 NotZOSparseMatrix) [inline]
```

**11.15.1.10 fspmv()** [6/12]

```
template<class Field , class SM >
void fspmv (
 const Field & F,
 const SM & A,
 typename Field::ConstElement_ptr x,
 typename Field::Element_ptr y,
 FieldCategories::GenericTag ,
 ZOSparseMatrix) [inline]
```

**11.15.1.11 fspmv()** [7/12]

```
template<class Field , class SM >
std::enable_if<!isSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (
 const Field & F,
 const SM & A,
 typename Field::ConstElement_ptr x,
 typename Field::Element_ptr y,
 FieldCategories::UnparametricTag ,
 ZOSparseMatrix) [inline]
```

**11.15.1.12 fspmv()** [8/12]

```
template<class Field , class SM >
std::enable_if< isSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (
 const Field & F,
 const SM & A,
 typename Field::ConstElement_ptr x,
 typename Field::Element_ptr y,
 FieldCategories::UnparametricTag ,
 ZOSparseMatrix) [inline]
```

**11.15.1.13 fspmv()** [9/12]

```
template<class Field , class SM >
void fspmv (
 const Field & F,
 const SM & A,
 typename Field::ConstElement_ptr x,
 typename Field::Element_ptr y,
 FieldCategories::ModularTag ,
 std::true_type) [inline]
```

**11.15.1.14 fspmm\_dispatch()** [1/2]

```
template<class Field , class SM , class FCat , class MZO >
std::enable_if<!(std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineInt
>::value||std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineInt
>::value)>::type fspmm_dispatch (
 const Field & F,
 const SM & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 FCat ,
 MZO) [inline]
```

**11.15.1.15 fspmm\_dispatch()** [2/2]

```
template<class Field , class SM , class FCat , class MZO >
std::enable_if< std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineInt
>::value||std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineInt
>::value >::type fspmm_dispatch (
 const Field & F,
 const SM & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 FCat ,
 MZO) [inline]
```

**11.15.1.16 fspmm()** [1/9]

```
template<class Field , class SM >
void fspmm (
 const Field & F,
 const SM & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 FieldCategories::GenericTag ,
 NotZOSparseMatrix) [inline]
```

**11.15.1.17 fspmm()** [2/9]

```
template<class Field , class SM >
std::enable_if< support_simd< typenameField::Element >::value >::type fspmm (
 const Field & F,
 const SM & A,
```



```

 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 FieldCategories::UnparametricTag ,
 NotZOSparseMatrix) [inline]

```

#### 11.15.1.18 fspmm() [3/9]

```

template<class Field , class SM >
std::enable_if<!support_simd< typenameField::Element >::value >::type fspmm (
 const Field & F,
 const SM & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 FieldCategories::UnparametricTag ,
 NotZOSparseMatrix) [inline]

```

#### 11.15.1.19 fspmm() [4/9]

```

template<class Field , class SM >
std::enable_if< support_simd< typenameField::Element >::value >::type fspmm (
 const Field & F,
 const SM & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 FieldCategories::ModularTag ,
 NotZOSparseMatrix) [inline]

```

#### 11.15.1.20 fspmm() [5/9]

```

template<class Field , class SM >
std::enable_if<!support_simd< typenameField::Element >::value >::type fspmm (
 const Field & F,
 const SM & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 FieldCategories::ModularTag ,
 NotZOSparseMatrix) [inline]

```

**11.15.1.21 fspmm() [6/9]**

```
template<class Field , class SM >
void fspmm (
 const Field & F,
 const SM & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 FieldCategories::GenericTag ,
 ZOSparseMatrix) [inline]
```

**11.15.1.22 fspmm() [7/9]**

```
template<class Field , class SM >
std::enable_if< support_simd< typenameField::Element >::value >::type fspmm (
 const Field & F,
 const SM & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 FieldCategories::UnparametricTag ,
 ZOSparseMatrix) [inline]
```

**11.15.1.23 fspmm() [8/9]**

```
template<class Field , class SM >
std::enable_if< !support_simd< typenameField::Element >::value >::type fspmm (
 const Field & F,
 const SM & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 FieldCategories::UnparametricTag ,
 ZOSparseMatrix) [inline]
```

**11.15.1.24 fspmm() [9/9]**

```
template<class Field , class SM >
void fspmm (
 const Field & F,
 const SM & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 FieldCategories::ModularTag ,
 ZOSparseMatrix) [inline]
```

**11.15.1.25 pfspmm\_dispatch()** [1/2]

```
template<class Field , class SM , class FCat , class MZO >
std::enable_if<!(std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineInt
>::value||std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineInt
>::value)>::type pfspmm_dispatch (
 const Field & F,
 const SM & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 FCat ,
 MZO) [inline]
```

**11.15.1.26 pfspmm\_dispatch()** [2/2]

```
template<class Field , class SM , class FCat , class MZO >
std::enable_if< std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineInt
>::value||std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineInt
>::value >::type pfspmm_dispatch (
 const Field & F,
 const SM & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 FCat ,
 MZO) [inline]
```

**11.15.1.27 pfspmm()** [1/9]

```
template<class Field , class SM >
void pfspmm (
 const Field & F,
 const SM & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 FieldCategories::GenericTag ,
 NotZOSparseMatrix) [inline]
```

**11.15.1.28 pfspmm()** [2/9]

```
template<class Field , class SM >
std::enable_if< support_simd< typenameField::Element >::value >::type pfspmm (
 const Field & F,
 const SM & A,
```

```

 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 FieldCategories::UnparametricTag ,
 NotZOSparseMatrix) [inline]

```

#### 11.15.1.29 pfspmm() [3/9]

```

template<class Field , class SM >
std::enable_if<!support_simd< typenameField::Element >::value >::type pfspmm (
 const Field & F,
 const SM & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 FieldCategories::UnparametricTag ,
 NotZOSparseMatrix) [inline]

```

#### 11.15.1.30 pfspmm() [4/9]

```

template<class Field , class SM >
std::enable_if< support_simd< typenameField::Element >::value >::type pfspmm (
 const Field & F,
 const SM & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 FieldCategories::ModularTag ,
 NotZOSparseMatrix) [inline]

```

#### 11.15.1.31 pfspmm() [5/9]

```

template<class Field , class SM >
std::enable_if<!support_simd< typenameField::Element >::value >::type pfspmm (
 const Field & F,
 const SM & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 FieldCategories::ModularTag ,
 NotZOSparseMatrix) [inline]

```

**11.15.1.32 pfspmm()** [6/9]

```
template<class Field , class SM >
void pfspmm (
 const Field & F,
 const SM & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 FieldCategories::GenericTag ,
 ZOSparseMatrix) [inline]
```

**11.15.1.33 pfspmm()** [7/9]

```
template<class Field , class SM >
std::enable_if< support_simd< typenameField::Element >::value >::type pfspmm (
 const Field & F,
 const SM & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 FieldCategories::UnparametricTag ,
 ZOSparseMatrix) [inline]
```

**11.15.1.34 pfspmm()** [8/9]

```
template<class Field , class SM >
std::enable_if< !support_simd< typenameField::Element >::value >::type pfspmm (
 const Field & F,
 const SM & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 FieldCategories::UnparametricTag ,
 ZOSparseMatrix) [inline]
```

**11.15.1.35 pfspmm()** [9/9]

```
template<class Field , class SM >
void pfspmm (
 const Field & F,
 const SM & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 FieldCategories::ModularTag ,
 ZOSparseMatrix) [inline]
```

**11.15.1.36 pfspmv()** [1/6]

```
template<class Field , class SM >
void pfspmv (
 const Field & F,
 const SM & A,
 typename Field::ConstElement_ptr x,
 typename Field::Element_ptr y,
 FieldCategories::GenericTag ,
 std::false_type) [inline]
```

**11.15.1.37 pfspmv()** [2/6]

```
template<class Field , class SM >
void pfspmv (
 const Field & F,
 const SM & A,
 typename Field::ConstElement_ptr x,
 typename Field::Element_ptr y,
 FieldCategories::UnparametricTag ,
 std::false_type) [inline]
```

**11.15.1.38 pfspmv()** [3/6]

```
template<class Field , class SM >
void pfspmv (
 const Field & F,
 const SM & A,
 typename Field::ConstElement_ptr x,
 typename Field::Element_ptr y,
 FieldCategories::ModularTag ,
 std::false_type) [inline]
```

**11.15.1.39 pfspmv()** [4/6]

```
template<class Field , class SM >
void pfspmv (
 const Field & F,
 const SM & A,
 typename Field::ConstElement_ptr x,
 typename Field::Element_ptr y,
 FieldCategories::GenericTag ,
 std::true_type) [inline]
```

**11.15.1.40 pfspmv()** [5/6]

```
template<class Field , class SM >
void pfspmv (
 const Field & F,
 const SM & A,
 typename Field::ConstElement_ptr x,
 typename Field::Element_ptr y,
 FieldCategories::UnparametricTag ,
 std::true_type) [inline]
```

**11.15.1.41 pfspmv()** [6/6]

```
template<class Field , class SM >
void pfspmv (
 const Field & F,
 const SM & A,
 typename Field::ConstElement_ptr x,
 typename Field::Element_ptr y,
 FieldCategories::ModularTag ,
 std::true_type) [inline]
```

**11.15.1.42 fspmv()** [10/12]

```
template<class Field , class SM >
std::enable_if< isSparseMatrixSimdFormat< Field, SM >::value &&support_simd< typenameField↵
::Element >::value >::type fspmv (
 const Field & F,
 const SM & A,
 typename Field::ConstElement_ptr x,
 typename Field::Element_ptr y,
 FieldCategories::UnparametricTag ,
 NotZOSparseMatrix) [inline]
```

**11.15.1.43 fspmv()** [11/12]

```
template<class Field , class SM >
std::enable_if< isSparseMatrixSimdFormat< Field, SM >::value &&support_simd< typenameField↵
::Element >::value >::type fspmv (
 const Field & F,
 const SM & A,
 typename Field::ConstElement_ptr x,
 typename Field::Element_ptr y,
 FieldCategories::ModularTag ,
 NotZOSparseMatrix) [inline]
```

**11.15.1.44 fspmv()** [12/12]

```
template<class Field , class SM >
std::enable_if< isSparseMatrixSimdFormat< Field, SM >::value &&support_simd< typenameField↵
::Element >::value >::type fspmv (
 const Field & F,
 const SM & A,
 typename Field::ConstElement_ptr x,
 typename Field::Element_ptr y,
 FieldCategories::UnparametricTag ,
 ZOSparseMatrix) [inline]
```

## 11.16 FFLAS::sparse\_details\_impl Namespace Reference

### Functions

- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::COO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::COO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::COO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, const int64_t kmax)`
- `template<class Field >`  
`void fspmm_simd_aligned (const Field &F, const Sparse< Field, SparseMatrix_t::COO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, const int64_t kmax)`
- `template<class Field >`  
`void fspmm_simd_unaligned (const Field &F, const Sparse< Field, SparseMatrix_t::COO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, const int64_t kmax)`
- `template<class Field >`  
`void fspmm_one (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmm_mone (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmm_one_simd_aligned (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_one_simd_unaligned (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_mone_simd_aligned (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_mone_simd_unaligned (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::COO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::COO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::COO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const uint64_t kmax)`
- `template<class Field >`  
`void fspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`



- `template<class Field >`  
`void fspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, const int64_t kmax)`
- `template<class Field >`  
`void pfspmm_one (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmm_mone (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmm_one (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmm_mone (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmv_task (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const index_t iStart, const index_t iStop, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const int64_t kmax)`
- `template<class Field >`  
`void pfspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`

- `template<class Field >`  
`void pfspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, index_t blockSize, typename Field::ConstElement_ptr x_, index_t ldx, typename Field::Element_ptr y_, index_t ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_simd_aligned (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_simd_unaligned (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, const int64_t kmax)`
- `template<class Field >`  
`void fspmm_one (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmm_mone (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmm_one_simd_aligned (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_one_simd_unaligned (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_mone_simd_aligned (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_mone_simd_unaligned (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const int64_t kmax)`
- `template<class Field >`  
`void fspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`

- `template<class Field >`  
`void fspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, size_t blockSize, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, size_t blockSize, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, size_t blockSize, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, const int64_t kmax)`
- `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, const int64_t kmax)`
- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const int64_t kmax)`
- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, const int64_t kmax)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const uint64_t kmax)`
- `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, size_t blockSize, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::GenericTag)`

- `template<class Field >`  
`void pfsppmm (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfsppmm (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, size_t blockSize, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfsppmm (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfsppmm (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, size_t blockSize, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, const int64_t kmax)`
- `template<class Field >`  
`void pfsppmm (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, const int64_t kmax)`
- `template<class Field, class Func >`  
`void pfsppmm_zo (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, Func &&func)`
- `template<class Field, class Func >`  
`void pfsppmm_zo (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, Func &&func)`
- `template<class Field >`  
`void pfsppmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfsppmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfsppmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const int64_t kmax)`
- `template<class Field >`  
`void pfsppmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfsppmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfsppmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfsppmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, const int64_t kmax)`
- `template<class Field >`  
`void fspmm_mone (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmm_one (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, size_t`

- blockSize, typename Field::ConstElement\_ptr x\_, int ldx, typename Field::Element\_ptr y\_, int ldy, FieldCategories::GenericTag)
- template<class Field >  
void fspmm\_mone (const Field &F, const Sparse< Field, SparseMatrix\_t::ELL\_ZO > &A, size\_t blockSize, typename Field::ConstElement\_ptr x\_, int ldx, typename Field::Element\_ptr y\_, int ldy, FieldCategories::UnparametricTag)
  - template<class Field >  
void fspmm\_one (const Field &F, const Sparse< Field, SparseMatrix\_t::ELL\_ZO > &A, size\_t blockSize, typename Field::ConstElement\_ptr x\_, int ldx, typename Field::Element\_ptr y\_, int ldy, FieldCategories::UnparametricTag)
  - template<class Field >  
void fspmm\_one\_simd\_aligned (const Field &F, const Sparse< Field, SparseMatrix\_t::ELL\_ZO > &A, size\_t blockSize, typename Field::ConstElement\_ptr x\_, int ldx, typename Field::Element\_ptr y\_, int ldy, FieldCategories::UnparametricTag)
  - template<class Field >  
void fspmm\_one\_simd\_unaligned (const Field &F, const Sparse< Field, SparseMatrix\_t::ELL\_ZO > &A, size\_t blockSize, typename Field::ConstElement\_ptr x\_, int ldx, typename Field::Element\_ptr y\_, int ldy, FieldCategories::UnparametricTag)
  - template<class Field >  
void fspmm\_mone\_simd\_aligned (const Field &F, const Sparse< Field, SparseMatrix\_t::ELL\_ZO > &A, size\_t blockSize, typename Field::ConstElement\_ptr x\_, int ldx, typename Field::Element\_ptr y\_, int ldy, FieldCategories::UnparametricTag)
  - template<class Field >  
void fspmm\_mone\_simd\_unaligned (const Field &F, const Sparse< Field, SparseMatrix\_t::ELL\_ZO > &A, size\_t blockSize, typename Field::ConstElement\_ptr x\_, int ldx, typename Field::Element\_ptr y\_, int ldy, FieldCategories::UnparametricTag)
  - template<class Field >  
void fspmv (const Field &F, const Sparse< Field, SparseMatrix\_t::ELL > &A, typename Field::ConstElement\_ptr x\_, typename Field::Element\_ptr y\_, FieldCategories::GenericTag)
  - template<class Field >  
void fspmv (const Field &F, const Sparse< Field, SparseMatrix\_t::ELL > &A, typename Field::ConstElement\_ptr x\_, typename Field::Element\_ptr y\_, FieldCategories::UnparametricTag)
  - template<class Field >  
void fspmv (const Field &F, const Sparse< Field, SparseMatrix\_t::ELL > &A, typename Field::ConstElement\_ptr x\_, typename Field::Element\_ptr y\_, const uint64\_t kmax)
  - template<class Field >  
void fspmv\_one (const Field &F, const Sparse< Field, SparseMatrix\_t::ELL\_ZO > &A, typename Field::ConstElement\_ptr x\_, typename Field::Element\_ptr y\_, FieldCategories::GenericTag)
  - template<class Field >  
void fspmv\_mone (const Field &F, const Sparse< Field, SparseMatrix\_t::ELL\_ZO > &A, typename Field::ConstElement\_ptr x\_, typename Field::Element\_ptr y\_, FieldCategories::GenericTag)
  - template<class Field >  
void fspmv\_one (const Field &F, const Sparse< Field, SparseMatrix\_t::ELL\_ZO > &A, typename Field::ConstElement\_ptr x\_, typename Field::Element\_ptr y\_, FieldCategories::UnparametricTag)
  - template<class Field >  
void fspmv\_mone (const Field &F, const Sparse< Field, SparseMatrix\_t::ELL\_ZO > &A, typename Field::ConstElement\_ptr x\_, typename Field::Element\_ptr y\_, FieldCategories::UnparametricTag)
  - template<class Field >  
void pfspmv (const Field &F, const Sparse< Field, SparseMatrix\_t::ELL\_simd > &A, typename Field::ConstElement\_ptr x\_, typename Field::Element\_ptr y\_, FieldCategories::GenericTag)
  - template<class Field >  
void pfspmv (const Field &F, const Sparse< Field, SparseMatrix\_t::ELL\_simd > &A, typename Field::ConstElement\_ptr x\_, typename Field::Element\_ptr y\_, FieldCategories::UnparametricTag)
  - template<class Field >  
void pfspmv (const Field &F, const Sparse< Field, SparseMatrix\_t::ELL\_simd > &A, typename Field::ConstElement\_ptr x\_, typename Field::Element\_ptr y\_, const uint64\_t kmax)

- `template<class Field >`  
`void pfspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv_simd (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_simd (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const uint64_t kmax)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const uint64_t kmax)`
- `template<class Field >`  
`void fspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_one_simd (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_mone_simd (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, uint64_t kmax)`
- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::GenericTag)`



- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, uint64_t kmax)`
- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, uint64_t kmax)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::HYB_ZO > &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, uint64_t kmax)`
- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::SELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::SELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::SELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const int64_t kmax)`
- `template<class Field >`  
`void pfspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::SELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::SELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::SELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::SELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::SELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv_simd (const Field &F, const Sparse< Field, SparseMatrix_t::SELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::SELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_simd (const Field &F, const Sparse< Field, SparseMatrix_t::SELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const uint64_t kmax)`

- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::SELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const uint64_t kmax)`
- `template<class Field >`  
`void fspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::SELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::SELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv_one_simd (const Field &F, const Sparse< Field, SparseMatrix_t::SELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_mone_simd (const Field &F, const Sparse< Field, SparseMatrix_t::SELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::SELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::SELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`

## 11.16.1 Function Documentation

### 11.16.1.1 fspmm() [1/15]

```
template<class Field >
void fspmm (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::COO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::GenericTag) [inline]
```

### 11.16.1.2 fspmm() [2/15]

```
template<class Field >
void fspmm (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::COO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::UnparametricTag) [inline]
```



**11.16.1.3 fspmm()** [3/15]

```
template<class Field >
void fspmm (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::COO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 const int64_t kmax) [inline]
```

**11.16.1.4 fspmm\_simd\_aligned()** [1/2]

```
template<class Field >
void fspmm_simd_aligned (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::COO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 const int64_t kmax) [inline]
```

**11.16.1.5 fspmm\_simd\_unaligned()** [1/2]

```
template<class Field >
void fspmm_simd_unaligned (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::COO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 const int64_t kmax) [inline]
```

**11.16.1.6 fspmm\_one()** [1/4]

```
template<class Field >
void fspmm_one (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::COO_ZO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.7 fspmm\_mone()** [1/4]

```
template<class Field >
void fspmm_mone (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::COO_ZO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.8 fspmm\_one\_simd\_aligned()** [1/3]

```
template<class Field >
void fspmm_one_simd_aligned (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::COO_ZO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.9 fspmm\_one\_simd\_unaligned()** [1/3]

```
template<class Field >
void fspmm_one_simd_unaligned (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::COO_ZO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.10 fspmm\_mone\_simd\_aligned()** [1/3]

```
template<class Field >
void fspmm_mone_simd_aligned (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::COO_ZO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.11 fspmm\_mone\_simd\_unaligned()** [1/3]

```
template<class Field >
void fspmm_mone_simd_unaligned (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::COO_ZO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.12 fspmv()** [1/21]

```
template<class Field >
void fspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::COO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.13 fspmv()** [2/21]

```
template<class Field >
void fspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::COO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.14 fspmv()** [3/21]

```
template<class Field >
void fspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::COO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 const uint64_t kmax) [inline]
```

**11.16.1.15 fspmv\_one()** [1/10]

```
template<class Field >
void fspmv_one (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::COO_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.16 fspmv\_mone()** [1/10]

```
template<class Field >
void fspmv_mone (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::COO_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.17 fspmv\_one()** [2/10]

```
template<class Field >
void fspmv_one (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::COO_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.18 fspmv\_mone()** [2/10]

```
template<class Field >
void fspmv_mone (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::COO_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.19 pfspmm()** [1/18]

```
template<class Field >
void pfspmm (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.20 pfspmm()** [2/18]

```
template<class Field >
void pfspmm (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.21 pfspmm()** [3/18]

```
template<class Field >
void pfspmm (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 const int64_t kmax) [inline]
```

**11.16.1.22 pfspmm\_one()** [1/2]

```
template<class Field >
void pfspmm_one (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.23 pfspmm\_mone()** [1/2]

```
template<class Field >
void pfspmm_mone (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.24 pfspmm\_one()** [2/2]

```
template<class Field >
void pfspmm_one (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.25 pfspmm\_mone()** [2/2]

```

template<class Field >
void pfspmm_mone (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::UnparametricTag) [inline]

```

**11.16.1.26 pfspmv()** [1/18]

```

template<class Field >
void pfspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::GenericTag) [inline]

```

**11.16.1.27 pfspmv\_task()**

```

template<class Field >
void pfspmv_task (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 const index_t iStart,
 const index_t iStop,
 FieldCategories::UnparametricTag) [inline]

```

**11.16.1.28 pfspmv()** [2/18]

```

template<class Field >
void pfspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]

```

**11.16.1.29 pfspmv()** [3/18]

```

template<class Field >
void pfspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 const int64_t kmax) [inline]

```

**11.16.1.30 pfspmv\_one()** [1/8]

```
template<class Field >
void pfspmv_one (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.31 pfspmv\_mone()** [1/8]

```
template<class Field >
void pfspmv_mone (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.32 pfspmv\_one()** [2/8]

```
template<class Field >
void pfspmv_one (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.33 pfspmv\_mone()** [2/8]

```
template<class Field >
void pfspmv_mone (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.34 fspmm()** [4/15]

```
template<class Field >
void fspmm (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.35 fspmm()** [5/15]

```

template<class Field >
void fspmm (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR > & A,
 index_t blockSize,
 typename Field::ConstElement_ptr x_,
 index_t ldx,
 typename Field::Element_ptr y_,
 index_t ldy,
 FieldCategories::UnparametricTag) [inline]

```

**11.16.1.36 fspmm\_simd\_aligned()** [2/2]

```

template<class Field >
void fspmm_simd_aligned (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::UnparametricTag) [inline]

```

**11.16.1.37 fspmm\_simd\_unaligned()** [2/2]

```

template<class Field >
void fspmm_simd_unaligned (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::UnparametricTag) [inline]

```

**11.16.1.38 fspmm()** [6/15]

```

template<class Field >
void fspmm (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 const int64_t kmax) [inline]

```



**11.16.139 fspmm\_one()** [2/4]

```
template<class Field >
void fspmm_one (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::GenericTag) [inline]
```

**11.16.140 fspmm\_mone()** [2/4]

```
template<class Field >
void fspmm_mone (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::GenericTag) [inline]
```

**11.16.141 fspmm\_one\_simd\_aligned()** [2/3]

```
template<class Field >
void fspmm_one_simd_aligned (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.142 fspmm\_one\_simd\_unaligned()** [2/3]

```
template<class Field >
void fspmm_one_simd_unaligned (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.143 fspmm\_mone\_simd\_aligned()** [2/3]

```
template<class Field >
void fspmm_mone_simd_aligned (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.144 fspmm\_mone\_simd\_unaligned()** [2/3]

```
template<class Field >
void fspmm_mone_simd_unaligned (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.145 fspmv()** [4/21]

```
template<class Field >
void fspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::GenericTag) [inline]
```

**11.16.146 fspmv()** [5/21]

```
template<class Field >
void fspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.147 fspmv()** [6/21]

```
template<class Field >
void fspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 const int64_t kmax) [inline]
```

**11.16.148 fspmv\_one()** [3/10]

```
template<class Field >
void fspmv_one (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::GenericTag) [inline]
```

**11.16.149 fspmv\_mone()** [3/10]

```
template<class Field >
void fspmv_mone (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::GenericTag) [inline]
```

**11.16.150 fspmv\_one()** [4/10]

```
template<class Field >
void fspmv_one (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.151 fspmv\_mone()** [4/10]

```
template<class Field >
void fspmv_mone (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.152 pfspmm()** [4/18]

```
template<class Field >
void pfspmm (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR_HYB > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 typename Field::Element_ptr y,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.53 pfspmm()** [5/18]

```

template<class Field >
void pfspmm (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR_HYB > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 FieldCategories::GenericTag) [inline]

```

**11.16.1.54 pfspmm()** [6/18]

```

template<class Field >
void pfspmm (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR_HYB > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 typename Field::Element_ptr y,
 FieldCategories::UnparametricTag) [inline]

```

**11.16.1.55 pfspmm()** [7/18]

```

template<class Field >
void pfspmm (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR_HYB > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 FieldCategories::UnparametricTag) [inline]

```

**11.16.1.56 pfspmm()** [8/18]

```

template<class Field >
void pfspmm (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR_HYB > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 typename Field::Element_ptr y,
 const int64_t kmax) [inline]

```

**11.16.1.57 pfspmm()** [9/18]

```
template<class Field >
void pfspmm (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR_HYB > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 const int64_t kmax) [inline]
```

**11.16.1.58 pfspmv()** [4/18]

```
template<class Field >
void pfspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR_HYB > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.59 pfspmv()** [5/18]

```
template<class Field >
void pfspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR_HYB > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.60 pfspmv()** [6/18]

```
template<class Field >
void pfspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR_HYB > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 const int64_t kmax) [inline]
```

**11.16.1.61 fspmm()** [7/15]

```
template<class Field >
void fspmm (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR_HYB > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.62 fspmm()** [8/15]

```
template<class Field >
void fspmm (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR_HYB > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.63 fspmm()** [9/15]

```
template<class Field >
void fspmm (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR_HYB > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 const int64_t kmax) [inline]
```

**11.16.1.64 fspmv()** [7/21]

```
template<class Field >
void fspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR_HYB > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.65 fspmv()** [8/21]

```
template<class Field >
void fspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR_HYB > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.66 fspmv()** [9/21]

```
template<class Field >
void fspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::CSR_HYB > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 const uint64_t kmax) [inline]
```

**11.16.1.67 pfspmm()** [10/18]

```
template<class Field >
void pfspmm (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 typename Field::Element_ptr y,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.68 pfspmm()** [11/18]

```
template<class Field >
void pfspmm (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.69 pfspmm()** [12/18]

```
template<class Field >
void pfspmm (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 typename Field::Element_ptr y,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.70 pfspmm()** [13/18]

```
template<class Field >
void pfspmm (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.71 pfspmm()** [14/18]

```
template<class Field >
void pfspmm (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 typename Field::Element_ptr y,
 const int64_t kmax) [inline]
```

**11.16.1.72 pfspmm()** [15/18]

```
template<class Field >
void pfspmm (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 const int64_t kmax) [inline]
```

**11.16.1.73 pfspmm\_zo()** [1/2]

```
template<class Field , class Func >
void pfspmm_zo (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 typename Field::Element_ptr y,
 Func && func) [inline]
```

**11.16.1.74 pfspmm\_zo()** [2/2]

```
template<class Field , class Func >
void pfspmm_zo (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 Func && func) [inline]
```



**11.16.1.75 pfspmv()** [7/18]

```
template<class Field >
void pfspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.76 pfspmv()** [8/18]

```
template<class Field >
void pfspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.77 pfspmv()** [9/18]

```
template<class Field >
void pfspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 const int64_t kmax) [inline]
```

**11.16.1.78 pfspmv\_one()** [3/8]

```
template<class Field >
void pfspmv_one (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.79 pfspmv\_mone()** [3/8]

```
template<class Field >
void pfspmv_mone (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.80 pfspmv\_one()** [4/8]

```
template<class Field >
void pfspmv_one (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.81 pfspmv\_mone()** [4/8]

```
template<class Field >
void pfspmv_mone (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.82 fspmm()** [10/15]

```
template<class Field >
void fspmm (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.83 fspmm()** [11/15]

```
template<class Field >
void fspmm (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.84 fspmm()** [12/15]

```
template<class Field >
void fspmm (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 const int64_t kmax) [inline]
```

**11.16.1.85 fspmm\_mone()** [3/4]

```
template<class Field >
void fspmm_mone (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.86 fspmm\_one()** [3/4]

```
template<class Field >
void fspmm_one (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.87 fspmm\_mone()** [4/4]

```
template<class Field >
void fspmm_mone (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.88 fspmm\_one() [4/4]**

```

template<class Field >
void fspmm_one (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::UnparametricTag) [inline]

```

**11.16.1.89 fspmm\_one\_simd\_aligned() [3/3]**

```

template<class Field >
void fspmm_one_simd_aligned (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::UnparametricTag) [inline]

```

**11.16.1.90 fspmm\_one\_simd\_unaligned() [3/3]**

```

template<class Field >
void fspmm_one_simd_unaligned (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::UnparametricTag) [inline]

```

**11.16.1.91 fspmm\_mone\_simd\_aligned() [3/3]**

```

template<class Field >
void fspmm_mone_simd_aligned (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::UnparametricTag) [inline]

```

**11.16.1.92 fspmm\_mone\_simd\_unaligned()** [3/3]

```
template<class Field >
void fspmm_mone_simd_unaligned (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x_,
 int ldx,
 typename Field::Element_ptr y_,
 int ldy,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.93 fspmv()** [10/21]

```
template<class Field >
void fspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.94 fspmv()** [11/21]

```
template<class Field >
void fspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.95 fspmv()** [12/21]

```
template<class Field >
void fspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 const uint64_t kmax) [inline]
```

**11.16.1.96 fspmv\_one()** [5/10]

```
template<class Field >
void fspmv_one (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.97 fspmv\_mone()** [5/10]

```
template<class Field >
void fspmv_mone (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.98 fspmv\_one()** [6/10]

```
template<class Field >
void fspmv_one (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.99 fspmv\_mone()** [6/10]

```
template<class Field >
void fspmv_mone (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.100 pfspmv()** [10/18]

```
template<class Field >
void pfspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_simd > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.101 pfspmv()** [11/18]

```
template<class Field >
void pfspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_simd > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.102 pfspmv()** [12/18]

```
template<class Field >
void pfspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_simd > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 const uint64_t kmax) [inline]
```

**11.16.1.103 pfspmv\_one()** [5/8]

```
template<class Field >
void pfspmv_one (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.104 pfspmv\_mone()** [5/8]

```
template<class Field >
void pfspmv_mone (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.105 pfspmv\_one()** [6/8]

```
template<class Field >
void pfspmv_one (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.106 pfspmv\_mone()** [6/8]

```
template<class Field >
void pfspmv_mone (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.107 fspmv()** [13/21]

```
template<class Field >
void fspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_simd > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.108 fspmv\_simd()** [1/4]

```
template<class Field >
void fspmv_simd (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_simd > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.109 fspmv()** [14/21]

```
template<class Field >
void fspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_simd > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.110 fspmv\_simd()** [2/4]

```
template<class Field >
void fspmv_simd (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_simd > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 const uint64_t kmax) [inline]
```

**11.16.1.111 fspmv()** [15/21]

```
template<class Field >
void fspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_simd > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 const uint64_t kmax) [inline]
```



**11.16.1.112 fspmv\_one()** [7/10]

```
template<class Field >
void fspmv_one (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.113 fspmv\_mone()** [7/10]

```
template<class Field >
void fspmv_mone (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.114 fspmv\_one()** [8/10]

```
template<class Field >
void fspmv_one (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.115 fspmv\_mone()** [8/10]

```
template<class Field >
void fspmv_mone (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.116 fspmv\_one\_simd()** [1/2]

```
template<class Field >
void fspmv_one_simd (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.117 fspmv\_mone\_simd()** [1/2]

```
template<class Field >
void fspmv_mone_simd (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.118 pfspmm()** [16/18]

```
template<class Field >
void pfspmm (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::HYB_ZO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.119 pfspmm()** [17/18]

```
template<class Field >
void pfspmm (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::HYB_ZO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.120 pfspmm()** [18/18]

```
template<class Field >
void pfspmm (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::HYB_ZO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 uint64_t kmax) [inline]
```

**11.16.1.121 pfspmv()** [13/18]

```
template<class Field >
void pfspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::HYB_ZO > & A,
 typename Field::ConstElement_ptr x,
 typename Field::Element_ptr y,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.122 pfspmv()** [14/18]

```
template<class Field >
void pfspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::HYB_ZO > & A,
 typename Field::ConstElement_ptr x,
 typename Field::Element_ptr y,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.123 pfspmv()** [15/18]

```
template<class Field >
void pfspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::HYB_ZO > & A,
 typename Field::ConstElement_ptr x,
 typename Field::Element_ptr y,
 uint64_t kmax) [inline]
```

**11.16.1.124 fspmm()** [13/15]

```
template<class Field >
void fspmm (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::HYB_ZO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.125 fspmm()** [14/15]

```
template<class Field >
void fspmm (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::HYB_ZO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.126 fspmm()** [15/15]

```
template<class Field >
void fspmm (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::HYB_ZO > & A,
 size_t blockSize,
 typename Field::ConstElement_ptr x,
 int ldx,
 typename Field::Element_ptr y,
 int ldy,
 uint64_t kmax) [inline]
```

**11.16.1.127 fspmv()** [16/21]

```
template<class Field >
void fspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::HYB_ZO > & A,
 typename Field::ConstElement_ptr x,
 typename Field::Element_ptr y,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.128 fspmv()** [17/21]

```
template<class Field >
void fspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::HYB_ZO > & A,
 typename Field::ConstElement_ptr x,
 typename Field::Element_ptr y,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.129 fspmv()** [18/21]

```
template<class Field >
void fspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::HYB_ZO > & A,
 typename Field::ConstElement_ptr x,
 typename Field::Element_ptr y,
 uint64_t kmax) [inline]
```

**11.16.1.130 pfspmv()** [16/18]

```
template<class Field >
void pfspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::SELL > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.131 pfspmv()** [17/18]

```
template<class Field >
void pfspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::SELL > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.132 pfspmv()** [18/18]

```
template<class Field >
void pfspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::SELL > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 const int64_t kmax) [inline]
```

**11.16.1.133 pfspmv\_one()** [7/8]

```
template<class Field >
void pfspmv_one (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::SELL_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.134 pfspmv\_mone()** [7/8]

```
template<class Field >
void pfspmv_mone (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::SELL_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.135 pfspmv\_one()** [8/8]

```
template<class Field >
void pfspmv_one (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::SELL_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.136 pfspmv\_mone()** [8/8]

```
template<class Field >
void pfspmv_mone (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::SELL_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.137 fspmv()** [19/21]

```
template<class Field >
void fspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::SELL > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.138 fspmv\_simd()** [3/4]

```
template<class Field >
void fspmv_simd (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::SELL > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.139 fspmv()** [20/21]

```
template<class Field >
void fspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::SELL > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.140 fspmv\_simd()** [4/4]

```
template<class Field >
void fspmv_simd (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::SELL > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 const uint64_t kmax) [inline]
```

**11.16.1.141 fspmv()** [21/21]

```
template<class Field >
void fspmv (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::SELL > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 const uint64_t kmax) [inline]
```

**11.16.1.142 fspmv\_one()** [9/10]

```
template<class Field >
void fspmv_one (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::SELL_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.143 fspmv\_mone()** [9/10]

```
template<class Field >
void fspmv_mone (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::SELL_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::GenericTag) [inline]
```

**11.16.1.144 fspmv\_one\_simd()** [2/2]

```
template<class Field >
void fspmv_one_simd (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::SELL_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.145 fspmv\_mone\_simd()** [2/2]

```
template<class Field >
void fspmv_mone_simd (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::SELL_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.146 fspmv\_one()** [10/10]

```
template<class Field >
void fspmv_one (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::SELL_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]
```

**11.16.1.147 fspmv\_mone()** [10/10]

```
template<class Field >
void fspmv_mone (
 const Field & F,
 const Sparse< Field, SparseMatrix_t::SELL_ZO > & A,
 typename Field::ConstElement_ptr x_,
 typename Field::Element_ptr y_,
 FieldCategories::UnparametricTag) [inline]
```

**11.17 FFLAS::StrategyParameter Namespace Reference****Data Structures**

- struct [Fixed](#)
- struct [Grain](#)
- struct [Threads](#)
- struct [ThreeD](#)
- struct [ThreeDAdaptive](#)
- struct [ThreeDInPlace](#)
- struct [TwoD](#)
- struct [TwoDAdaptive](#)

**11.18 FFLAS::StructureHelper Namespace Reference**

[StructureHelper](#) for ftrsm.

**Data Structures**

- struct [Hybrid](#)
- struct [Iterative](#)
- struct [Recursive](#)

**11.18.1 Detailed Description**

[StructureHelper](#) for ftrsm.



## 11.19 FFLAS::vectorised Namespace Reference

### Namespaces

- namespace [unswitch](#)

### Data Structures

- struct [HelperMod](#)
- struct [HelperMod](#)< [Field](#), [ElementCategories::MachineIntTag](#) >
- struct [HelperMod](#)< [Field](#), [FFLAS::ElementCategories::ArbitraryPrecIntTag](#) >
- struct [HelperMod](#)< [Field](#), [FFLAS::ElementCategories::FixedPrecIntTag](#) >
- struct [HelperMod](#)< [Field](#), [FFLAS::ElementCategories::MachineFloatTag](#) >

### Functions

- [template<class SimdT , class Element , bool positive>](#)  
[std::enable\\_if< is\\_simd< SimdT >::value, void >::type](#) [VEC\\_ADD](#) ([SimdT](#) &C, [SimdT](#) &A, [SimdT](#) &B, [SimdT](#) &Q, [SimdT](#) &T, [SimdT](#) &P, [SimdT](#) &NEGP, [SimdT](#) &MIN, [SimdT](#) &MAX)
- [template<bool positive, class Element , class T1 , class T2 >](#)  
[std::enable\\_if< FFLAS::support\\_simd\\_add< Element >::value, void >::type](#) [addp](#) ([Element](#) \*T, [const](#) [Element](#) \*TA, [const](#) [Element](#) \*TB, [size\\_t](#) n, [Element](#) p, [T1](#) min\_, [T2](#) max\_)
- [template<class SimdT , class Element , bool positive>](#)  
[std::enable\\_if< is\\_simd< SimdT >::value, void >::type](#) [VEC\\_SUB](#) ([SimdT](#) &C, [SimdT](#) &A, [SimdT](#) &B, [SimdT](#) &Q, [SimdT](#) &T, [SimdT](#) &P, [SimdT](#) &NEGP, [SimdT](#) &MIN, [SimdT](#) &MAX)
- [template<bool positive, class Element , class T1 , class T2 >](#)  
[std::enable\\_if< FFLAS::support\\_simd\\_add< Element >::value, void >::type](#) [subp](#) ([Element](#) \*T, [const](#) [Element](#) \*TA, [const](#) [Element](#) \*TB, [const](#) [size\\_t](#) n, [const](#) [Element](#) p, [const](#) [T1](#) min\_, [const](#) [T2](#) max\_)
- [template<class Element >](#)  
[std::enable\\_if< FFLAS::support\\_simd\\_add< Element >::value, void >::type](#) [add](#) ([Element](#) \*T, [const](#) [Element](#) \*TA, [const](#) [Element](#) \*TB, [size\\_t](#) n)
- [template<class Element >](#)  
[std::enable\\_if< FFLAS::support\\_simd\\_add< Element >::value, void >::type](#) [sub](#) ([Element](#) \*T, [const](#) [Element](#) \*TA, [const](#) [Element](#) \*TB, [size\\_t](#) n)
- [template<class Field >](#)  
[std::enable\\_if< FFLAS::support\\_fast\\_mod< typenameField::Element >::value, void >::type](#) [axpyp](#) ([const](#) [Field](#) &F, [const](#) [typenameField::Element](#) a, [typenameField::ConstElement\\_ptr](#) X, [typenameField::Element\\_ptr](#) Y, [const](#) [size\\_t](#) n)
- [template<class Field >](#)  
[std::enable\\_if< FFLAS::support\\_fast\\_mod< typenameField::Element >::value, void >::type](#) [axpyp](#) ([const](#) [Field](#) &F, [const](#) [typenameField::Element](#) a, [typenameField::ConstElement\\_ptr](#) X, [typenameField::Element\\_ptr](#) Y, [const](#) [size\\_t](#) n, [const](#) [size\\_t](#) incX, [const](#) [size\\_t](#) incY)
- [template<class T >](#)  
[std::enable\\_if<!std::is\\_integral< T >::value, T >::type](#) [reduce](#) ([T](#) A, [T](#) B)
- [template<class T >](#)  
[std::enable\\_if< std::is\\_integral< T >::value, T >::type](#) [reduce](#) ([T](#) A, [T](#) B)
- [template<>](#) [Givaro::Integer](#) [reduce](#) ([Givaro::Integer](#) A, [Givaro::Integer](#) B)
- [float](#) [reduce](#) ([float](#) A, [float](#) B, [float](#) invB, [float](#) min, [float](#) max)
- [double](#) [reduce](#) ([double](#) A, [double](#) B, [double](#) invB, [double](#) min, [double](#) max)
- [int64\\_t](#) [reduce](#) ([int64\\_t](#) A, [int64\\_t](#) p, [double](#) invp, [double](#) min, [double](#) max, [int64\\_t](#) pow50rem)
- [template<class Field >](#)  
[Field::Element](#) [reduce](#) ([typenameField::Element](#) A, [HelperMod](#)< [Field](#), [ElementCategories::MachineIntTag](#) > &H)

- `template<class Field >`  
`Field::Element reduce (typename Field::Element A, HelperMod< Field, ElementCategories::MachineFloatTag > &H)`
- `template<class Field >`  
`Field::Element reduce (typename Field::Element A, HelperMod< Field, ElementCategories::ArbitraryPrecIntTag > &H)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type modp (const Field &F, typename Field::ConstElement_ptr U, const size_t &n, typename Field::Element_ptr T)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type modp (const Field &F, typename Field::ConstElement_ptr U, const size_t &n, const size_t &incX, typename Field::Element_ptr T)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type scalp (const Field &F, typename Field::Element_ptr T, const typename Field::Element alpha, typename Field::ConstElement_ptr U, const size_t n)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type scalp (const Field &F, typename Field::Element_ptr T, const typename Field::Element alpha, typename Field::ConstElement_ptr U, const size_t n, const size_t &incX)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type scalp (const Field &F, typename Field::Element_ptr T, const typename Field::Element alpha, typename Field::ConstElement_ptr U, const size_t n, const size_t &incX, const size_t &incY)`

## 11.19.1 Function Documentation

### 11.19.1.1 VEC\_ADD()

```
template<class SimdT , class Element , bool positive>
std::enable_if< is_simd< SimdT >::value, void >::type VEC_ADD (
 SimdT & C,
 SimdT & A,
 SimdT & B,
 SimdT & Q,
 SimdT & T,
 SimdT & P,
 SimdT & NEGP,
 SimdT & MIN,
 SimdT & MAX) [inline]
```

### 11.19.1.2 addp()

```
template<bool positive, class Element , class T1 , class T2 >
std::enable_if< FFLAS::support_simd_add< Element >::value, void >::type addp (
 Element * T,
 const Element * TA,
 const Element * TB,
 size_t n,
 Element p,
 T1 min_,
 T2 max_) [inline]
```

### 11.19.1.3 VEC\_SUB()

```
template<class SimdT , class Element , bool positive>
std::enable_if< is_simd< SimdT >::value, void >::type VEC_SUB (
 SimdT & C,
 SimdT & A,
 SimdT & B,
 SimdT & Q,
 SimdT & T,
 SimdT & P,
 SimdT & NEGP,
 SimdT & MIN,
 SimdT & MAX) [inline]
```

### 11.19.1.4 subp()

```
template<bool positive, class Element , class T1 , class T2 >
std::enable_if< FFLAS::support_simd_add< Element >::value, void >::type subp (
 Element * T,
 const Element * TA,
 const Element * TB,
 const size_t n,
 const Element p,
 const T1 min_,
 const T2 max_) [inline]
```

### 11.19.1.5 add()

```
template<class Element >
std::enable_if< FFLAS::support_simd_add< Element >::value, void >::type add (
 Element * T,
 const Element * TA,
 const Element * TB,
 size_t n) [inline]
```

### 11.19.1.6 sub()

```
template<class Element >
std::enable_if< FFLAS::support_simd_add< Element >::value, void >::type sub (
 Element * T,
 const Element * TA,
 const Element * TB,
 size_t n) [inline]
```

### 11.19.1.7 axpyp() [1/2]

```
template<class Field >
std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type axpyp
(
 const Field & F,
 const typenameField::Element a,
 typenameField::ConstElement_ptr X,
 typenameField::Element_ptr Y,
 const size_t n) [inline]
```

**11.19.1.8 axpyp()** [2/2]

```
template<class Field >
std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type axpyp
(
 const Field & F,
 const typename Field::Element a,
 typename Field::ConstElement_ptr X,
 typename Field::Element_ptr Y,
 const size_t n,
 const size_t incX,
 const size_t incY) [inline]
```

**11.19.1.9 reduce()** [1/9]

```
template<class T >
std::enable_if< !std::is_integral< T >::value, T >::type reduce (
 T A,
 T B) [inline]
```

**11.19.1.10 reduce()** [2/9]

```
template<class T >
std::enable_if< std::is_integral< T >::value, T >::type reduce (
 T A,
 T B) [inline]
```

**11.19.1.11 reduce()** [3/9]

```
template<>
Givaro::Integer reduce (
 Givaro::Integer A,
 Givaro::Integer B) [inline]
```

**11.19.1.12 reduce()** [4/9]

```
float reduce (
 float A,
 float B,
 float invB,
 float min,
 float max) [inline]
```

**11.19.1.13 reduce()** [5/9]

```
double reduce (
 double A,
 double B,
 double invB,
 double min,
 double max) [inline]
```

**11.19.1.14 reduce()** [6/9]

```
int64_t reduce (
 int64_t A,
 int64_t p,
 double invp,
 double min,
 double max,
 int64_t pow50rem) [inline]
```

**11.19.1.15 reduce()** [7/9]

```
template<class Field >
Field::Element reduce (
 typename Field::Element A,
 HelperMod< Field, ElementCategories::MachineIntTag > & H) [inline]
```

**11.19.1.16 reduce()** [8/9]

```
template<class Field >
Field::Element reduce (
 typename Field::Element A,
 HelperMod< Field, ElementCategories::MachineFloatTag > & H) [inline]
```

**11.19.1.17 reduce()** [9/9]

```
template<class Field >
Field::Element reduce (
 typename Field::Element A,
 HelperMod< Field, ElementCategories::ArbitraryPrecIntTag > & H) [inline]
```

**11.19.1.18 modp()** [1/2]

```
template<class Field >
std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type modp (
 const Field & F,
 typename Field::ConstElement_ptr U,
 const size_t & n,
 typename Field::Element_ptr T) [inline]
```

**11.19.1.19 modp()** [2/2]

```
template<class Field >
std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type modp (
 const Field & F,
 typename Field::ConstElement_ptr U,
 const size_t & n,
 const size_t & incX,
 typename Field::Element_ptr T) [inline]
```

**11.19.1.20 scalp() [1/3]**

```
template<class Field >
std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type scalp
(
 const Field & F,
 typename Field::Element_ptr T,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr U,
 const size_t n) [inline]
```

**11.19.1.21 scalp() [2/3]**

```
template<class Field >
std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type scalp
(
 const Field & F,
 typename Field::Element_ptr T,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr U,
 const size_t n,
 const size_t & incX) [inline]
```

**11.19.1.22 scalp() [3/3]**

```
template<class Field >
std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type scalp
(
 const Field & F,
 typename Field::Element_ptr T,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr U,
 const size_t n,
 const size_t & incX,
 const size_t & incY) [inline]
```

**11.20 FFLAS::vectorised::unswitch Namespace Reference****Functions**

- template<class Field >  
std::enable\_if< !FFLAS::support\_simd\_mod< typenameField::Element >::value &&FFLAS::support\_fast\_mod< typenameField::Element >::value, void >::type axypyp (const Field &F, const typename Field::Element a, typename Field::ConstElement\_ptr X, typename Field::Element\_ptr Y, const size\_t n, HelperMod< Field > &H)
- template<class Field >  
std::enable\_if< FFLAS::support\_fast\_mod< typenameField::Element >::value, void >::type axypyp (const Field &F, const typename Field::Element a, typename Field::ConstElement\_ptr X, typename Field::Element\_ptr Y, const size\_t n, const size\_t incX, const size\_t incY, HelperMod< Field > &H)

- `template<class Field >`  
`std::enable_if<!FFLAS::support_simd_mod< typenameField::Element >::value &&FFLAS::support_fast_mod<`  
`typenameField::Element >::value, void >::type modp (const Field &F, typename Field::ConstElement_ptr U,`  
`const size_t &n, typename Field::Element_ptr T, HelperMod< Field > &H)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type modp`  
`(const Field &F, typename Field::ConstElement_ptr U, const size_t &n, const size_t &incX, typename`  
`Field::Element_ptr T, HelperMod< Field > &H)`
- `template<class Field >`  
`std::enable_if<!FFLAS::support_simd_mod< typenameField::Element >::value &&FFLAS::support_fast_mod<`  
`typenameField::Element >::value, void >::type scalp (const Field &F, typename Field::Element_ptr T, const`  
`typename Field::Element alpha, typename Field::ConstElement_ptr U, const size_t n, HelperMod< Field >`  
`&H)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type scalp`  
`(const Field &F, typename Field::Element_ptr T, const typename Field::Element alpha, typename`  
`Field::ConstElement_ptr U, const size_t n, const size_t &incX, HelperMod< Field > &H)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type scalp`  
`(const Field &F, typename Field::Element_ptr T, const typename Field::Element alpha, typename`  
`Field::ConstElement_ptr U, const size_t n, const size_t &incX, const size_t &incY, HelperMod< Field >`  
`&H)`

## 11.20.1 Function Documentation

### 11.20.1.1 axpyp() [1/2]

```
template<class Field >
std::enable_if<!FFLAS::support_simd_mod< typenameField::Element >::value &&FFLAS::support_fast_mod<
typenameField::Element >::value, void >::type axpyp (
 const Field & F,
 const typename Field::Element a,
 typename Field::ConstElement_ptr X,
 typename Field::Element_ptr Y,
 const size_t n,
 HelperMod< Field > & H) [inline]
```

### 11.20.1.2 axpyp() [2/2]

```
template<class Field >
std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type axpyp
(
 const Field & F,
 const typename Field::Element a,
 typename Field::ConstElement_ptr X,
 typename Field::Element_ptr Y,
 const size_t n,
 const size_t incX,
 const size_t incY,
 HelperMod< Field > & H) [inline]
```

**11.20.1.3 modp()** [1/2]

```
template<class Field >
std::enable_if<!FFLAS::support_simd_mod< typenameField::Element >::value &&FFLAS::support_fast_mod<
typenameField::Element >::value, void >::type modp (
 const Field & F,
 typename Field::ConstElement_ptr U,
 const size_t & n,
 typename Field::Element_ptr T,
 HelperMod< Field > & H) [inline]
```

**11.20.1.4 modp()** [2/2]

```
template<class Field >
std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type modp (
 const Field & F,
 typename Field::ConstElement_ptr U,
 const size_t & n,
 const size_t & incX,
 typename Field::Element_ptr T,
 HelperMod< Field > & H) [inline]
```

**11.20.1.5 scalp()** [1/3]

```
template<class Field >
std::enable_if<!FFLAS::support_simd_mod< typenameField::Element >::value &&FFLAS::support_fast_mod<
typenameField::Element >::value, void >::type scalp (
 const Field & F,
 typename Field::Element_ptr T,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr U,
 const size_t n,
 HelperMod< Field > & H) [inline]
```

**11.20.1.6 scalp()** [2/3]

```
template<class Field >
std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type scalp
(
 const Field & F,
 typename Field::Element_ptr T,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr U,
 const size_t n,
 const size_t & incX,
 HelperMod< Field > & H) [inline]
```



11.20.1.7 `scalp()` [3/3]

```
template<class Field >
std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type scalp
(
 const Field & F,
 typename Field::Element_ptr T,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr U,
 const size_t n,
 const size_t & incX,
 const size_t & incY,
 HelperMod< Field > & H) [inline]
```

## 11.21 FFPACK Namespace Reference

Finite Field **PACK** Set of elimination based routines for dense linear algebra.

### Namespaces

- namespace [Protected](#)

### Data Structures

- class [callLUdivine\\_small](#)
- class [callLUdivine\\_small< double >](#)
- class [callLUdivine\\_small< float >](#)
- class [CharpolyFailed](#)
- class [CheckerImplem\\_charpoly](#)
- class [CheckerImplem\\_charpoly< Givaro::ZRing< Givaro::Integer >, Polynomial >](#)
- class [CheckerImplem\\_Det](#)
- class [CheckerImplem\\_invert](#)
- class [CheckerImplem\\_PLUQ](#)
- class [Failure](#)
  - A precondition failed.*
- struct [rns\\_double](#)
- struct [rns\\_double\\_elt](#)
- struct [rns\\_double\\_elt\\_cstptr](#)
- struct [rns\\_double\\_elt\\_ptr](#)
- struct [rns\\_double\\_extended](#)
- class [RNSInteger](#)
- class [RNSIntegerMod](#)
- class [rnsRandIter](#)

## Typedefs

- `template<class Field >`  
`using Checker_PLUQ = FFLAS::Checker_Empty< Field >`
- `template<class Field >`  
`using Checker_Det = FFLAS::Checker_Empty< Field >`
- `template<class Field >`  
`using Checker_invert = FFLAS::Checker_Empty< Field >`
- `template<class Field , class Polynomial >`  
`using Checker_charpoly = FFLAS::Checker_Empty< Field >`
- `template<class Field >`  
`using ForceCheck_PLUQ = CheckerImplem_PLUQ< Field >`
- `template<class Field >`  
`using ForceCheck_Det = CheckerImplem_Det< Field >`
- `template<class Field >`  
`using ForceCheck_invert = CheckerImplem_invert< Field >`
- `template<class Field , class Polynomial >`  
`using ForceCheck_charpoly = CheckerImplem_charpoly< Field, Polynomial >`

## Functions

- `void LAPACKPerm2MathPerm (size_t *MathP, const size_t *LapackP, const size_t N)`  
*Conversion of a permutation from LAPACK format to Math format.*
- `void MathPerm2LAPACKPerm (size_t *LapackP, const size_t *MathP, const size_t N)`  
*Conversion of a permutation from Maths format to LAPACK format.*
- `template<class Field >`  
`void applyP (const Field &F, const FFLAS::FFLAS_SIDE Side, const FFLAS::FFLAS_TRANSPOSE Trans, const size_t M, const size_t ibeg, const size_t iend, typename Field::Element_ptr A, const size_t lda, const size_t *P)`  
*Computes  $P1 \times \text{Diag}(I_R, P2)$  where  $P1$  is a LAPACK and  $P2$  a LAPACK permutation and store the result in  $P1$  as a LAPACK permutation.*
- `template<class Field >`  
`void applyP (const Field &F, const FFLAS::FFLAS_SIDE Side, const FFLAS::FFLAS_TRANSPOSE Trans, const size_t m, const size_t ibeg, const size_t iend, typename Field::Element_ptr A, const size_t lda, const size_t *P, const FFLAS::ParSeqHelper::Sequential seq)`
- `template<class Field , class Cut , class Param >`  
`void applyP (const Field &F, const FFLAS::FFLAS_SIDE Side, const FFLAS::FFLAS_TRANSPOSE Trans, const size_t m, const size_t ibeg, const size_t iend, typename Field::Element_ptr A, const size_t lda, const size_t *P, const FFLAS::ParSeqHelper::Parallel< Cut, Param > par)`
- `template<class Field >`  
`void MonotonicApplyP (const Field &F, const FFLAS::FFLAS_SIDE Side, const FFLAS::FFLAS_TRANSPOSE Trans, const size_t M, const size_t ibeg, const size_t iend, typename Field::Element_ptr A, const size_t lda, const size_t *P, const size_t R)`  
*Apply a R-monotonically increasing permutation P, to the matrix A.*
- `template<class Field >`  
`void fgetrs (const Field &F, const FFLAS::FFLAS_SIDE Side, const size_t M, const size_t N, const size_t R, typename Field::Element_ptr A, const size_t lda, const size_t *P, const size_t *Q, typename Field::Element_ptr B, const size_t ldb, int *info)`  
*Solve the system  $AX = B$  or  $XA = B$ .*
- `template<class Field >`  
`Field::Element_ptr fgetrs (const Field &F, const FFLAS::FFLAS_SIDE Side, const size_t M, const size_t N, const size_t NRHS, const size_t R, typename Field::Element_ptr A, const size_t lda, const size_t *P, const size_t *Q, typename Field::Element_ptr X, const size_t idx, typename Field::ConstElement_ptr B, const size_t ldb, int *info)`  
*Solve the system  $A X = B$  or  $X A = B$ .*

- `template<class Field >`  
`size_t fgesv (const Field &F, const FFLAS::FFLAS_SIDE Side, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb, int *info)`  
*Square system solver.*
- `template<class Field >`  
`size_t fgesv (const Field &F, const FFLAS::FFLAS_SIDE Side, const size_t M, const size_t N, const size_t NRHS, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr X, const size_t ldx, typename Field::ConstElement_ptr B, const size_t ldb, int *info)`  
*Rectangular system solver.*
- `template<class Field >`  
`void ftrtri (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG Diag, const size_t N, typename Field::Element_ptr A, const size_t lda, const size_t threshold=__FFLASFFPACK_FTRTRI_THRESHOLD)`  
*Compute the inverse of a triangular matrix.*
- `template<class Field >`  
`void trinv_left (const Field &F, const size_t N, typename Field::ConstElement_ptr L, const size_t ldl, typename Field::Element_ptr X, const size_t ldx)`
- `template<class Field >`  
`void ftrtm (const Field &F, const FFLAS::FFLAS_SIDE side, const FFLAS::FFLAS_DIAG diag, const size_t N, typename Field::Element_ptr A, const size_t lda)`  
*Compute the product of two triangular matrices of opposite shape.*
- `template<class Field >`  
`void ftrstr (const Field &F, const FFLAS::FFLAS_SIDE side, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG diagA, const FFLAS::FFLAS_DIAG diagB, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb, const size_t threshold=__FFLASFFPACK_FTRSTR_THRESHOLD)`  
*Solve a triangular system with a triangular right hand side of the same shape.*
- `template<class Field >`  
`void ftrssyr2k (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG diagA, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb, const size_t threshold=__FFLASFFPACK_FTRSSYR2K_THRESHOLD)`  
*Solve a triangular system in a symmetric sum: find B upper/lower triangular such that  $A^T B + B^T A = C$  where C is symmetric.*
- `template<class Field >`  
`bool fsytrf (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t lda, const size_t threshold=__FFLASFFPACK_FSYTRF_THRESHOLD)`  
*Triangular factorization of symmetric matrices.*
- `template<class Field >`  
`bool fsytrf (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t lda, const FFLAS::ParSeqHelper::Sequential seq, const size_t threshold=__FFLASFFPACK_FSYTRF_THRESHOLD)`
- `template<class Field, class Cut, class Param >`  
`bool fsytrf (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t lda, const FFLAS::ParSeqHelper::Parallel< Cut, Param > par, const size_t threshold=__FFLASFFPACK_FSYTRF_THRESHOLD)`
- `template<class Field >`  
`bool fsytrf_nonunit (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr D, const size_t incD, const size_t threshold=__FFLASFFPACK_FSYTRF_THRESHOLD)`  
*Triangular factorization of symmetric matrices.*
- `template<class Field >`  
`size_t PLUQ (const Field &F, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q)`  
*Compute a PLUQ factorization of the given matrix.*
- `template<class Field >`  
`size_t pPLUQ (const Field &F, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q)`

- `template<class Field >`  
`size_t PLUQ (const Field &F, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q, const FFLAS::ParSeqHelper::Sequential &PSHelper, size_t BCThreshold=__FFLASFFPACK_PLUQ_THRESHOLD)`
- `template<class Field , class Cut , class Param >`  
`size_t PLUQ (const Field &F, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q, const FFLAS::ParSeqHelper::Parallel< Cut, Param > &PSHelper)`
- `template<class Field >`  
`size_t LUdivine (const Field &F, const FFLAS::FFLAS_DIAG Diag, const FFLAS::FFLAS_TRANSPOSE trans, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const FFPACK_LU_TAG LuTag=FpackSlabRecursive, const size_t cutoff=__FFLASFFPACK_LUDIVINE_THRESHOLD)`  
*Compute the CUP or PLE factorization of the given matrix.*
- `template<class Field >`  
`size_t ColumnEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, bool transform=false, const FFPACK_LU_TAG LuTag=FpackSlabRecursive)`  
*Compute the Column Echelon form of the input matrix in-place.*
- `template<class Field >`  
`size_t pColumnEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, bool transform=false, size_t numthreads=0, const FFPACK_LU_TAG LuTag=FpackTileRecursive)`
- `template<class Field , class PSHelper >`  
`size_t ColumnEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const bool transform, const FFPACK_LU_TAG LuTag, const PSHelper &psH)`
- `template<class Field >`  
`size_t RowEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const bool transform=false, const FFPACK_LU_TAG LuTag=FpackSlabRecursive)`  
*Compute the Row Echelon form of the input matrix in-place.*
- `template<class Field >`  
`size_t pRowEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const bool transform=false, size_t numthreads=0, const FFPACK_LU_TAG LuTag=FpackTileRecursive)`
- `template<class Field , class PSHelper >`  
`size_t RowEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const bool transform, const FFPACK_LU_TAG LuTag, const PSHelper &psH)`
- `template<class Field >`  
`size_t ReducedColumnEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const bool transform=false, const FFPACK_LU_TAG LuTag=FpackSlabRecursive)`  
*Compute the Reduced Column Echelon form of the input matrix in-place.*
- `template<class Field >`  
`size_t pReducedColumnEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const bool transform=false, size_t numthreads=0, const FFPACK_LU_TAG LuTag=FpackTileRecursive)`
- `template<class Field , class PSHelper >`  
`size_t ReducedColumnEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const bool transform, const FFPACK_LU_TAG LuTag, const PSHelper &psH)`
- `template<class Field >`  
`size_t ReducedRowEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const bool transform=false, const FFPACK_LU_TAG LuTag=FpackSlabRecursive)`  
*Compute the Reduced Row Echelon form of the input matrix in-place.*

- `template<class Field >`  
`size_t pReducedRowEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const bool transform=false, size_t numthreads=0, const FFPACK_LU_TAG LuTag=FpackTileRecursive)`
- `template<class Field , class PSHelper >`  
`size_t ReducedRowEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const bool transform, const FFPACK_LU_TAG LuTag, const PSHelper &psH)`
- `template<class Field >`  
`Field::Element_ptr Invert (const Field &F, const size_t M, typename Field::Element_ptr A, const size_t lda, int &>nullity)`  
*Invert the given matrix in place or computes its nullity if it is singular.*
- `template<class Field >`  
`Field::Element_ptr Invert (const Field &F, const size_t M, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr X, const size_t idx, int &>nullity)`  
*Invert the given matrix or computes its nullity if it is singular.*
- `template<class Field >`  
`Field::Element_ptr Invert2 (const Field &F, const size_t M, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr X, const size_t idx, int &>nullity)`  
*Invert the given matrix or computes its nullity if it is singular.*
- `template<class PolRing >`  
`std::list< typename PolRing::Element > & CharPoly (const PolRing &R, std::list< typename PolRing::Element > &charp, const size_t N, typename PolRing::Domain_t::Element_ptr A, const size_t lda, typename PolRing::Domain_t::RandIter &G, const FFPACK_CHARPOLY_TAG CharpTag=FpackAuto, const size_t degree=__FFLASFFPACK_ARITHPROG_THRESHOLD)`  
*Compute the characteristic polynomial of the matrix A.*
- `template<class PolRing >`  
`PolRing::Element & CharPoly (const PolRing &R, typename PolRing::Element &charp, const size_t N, typename PolRing::Domain_t::Element_ptr A, const size_t lda, typename PolRing::Domain_t::RandIter &G, const FFPACK_CHARPOLY_TAG CharpTag=FpackAuto, const size_t degree=__FFLASFFPACK_ARITHPROG_THRESHOLD)`  
*Compute the characteristic polynomial of the matrix A.*
- `template<class PolRing >`  
`PolRing::Element & CharPoly (const PolRing &R, typename PolRing::Element &charp, const size_t N, typename PolRing::Domain_t::Element_ptr A, const size_t lda, const FFPACK_CHARPOLY_TAG CharpTag=FpackAuto, const size_t degree=__FFLASFFPACK_ARITHPROG_THRESHOLD)`  
*Compute the characteristic polynomial of the matrix A.*
- `template<class Field , class Polynomial >`  
`Polynomial & MinPoly (const Field &F, Polynomial &minP, const size_t N, typename Field::ConstElement_ptr A, const size_t lda)`  
*Compute the minimal polynomial of the matrix A.*
- `template<class Field , class Polynomial , class RandIter >`  
`Polynomial & MinPoly (const Field &F, Polynomial &minP, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, RandIter &G)`  
*Compute the minimal polynomial of the matrix A.*
- `template<class Field , class Polynomial >`  
`Polynomial & MatVecMinPoly (const Field &F, Polynomial &minP, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr v, const size_t incv)`  
*Compute the minimal polynomial of the matrix A and a vector v, namely the first linear dependency relation in the Krylov basis  $(v, Av, \dots, A^N v)$ .*
- `template<class Field >`  
`size_t Rank (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda)`  
*Computes the rank of the given matrix using a PLUQ factorization.*
- `template<class Field >`  
`size_t pRank (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t numthreads=0)`

- `template<class Field , class PSHelper >`  
`size_t Rank (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t Ida, const PSHelper &psH)`
- `template<class Field >`  
`bool IsSingular (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t Ida)`  
*Returns true if the given matrix is singular.*
- `template<class Field >`  
`Field::Element & Det (const Field &F, typename Field::Element &det, const size_t N, typename Field::Element_ptr A, const size_t Ida, size_t *P=NULL, size_t *Q=NULL)`  
*Returns the determinant of the given square matrix.*
- `template<class Field >`  
`Field::Element & pDet (const Field &F, typename Field::Element &det, const size_t N, typename Field::Element_ptr A, const size_t Ida, size_t numthreads=0, size_t *P=NULL, size_t *Q=NULL)`
- `template<class Field , class PSHelper >`  
`Field::Element & Det (const Field &F, typename Field::Element &det, const size_t N, typename Field::Element_ptr A, const size_t Ida, const PSHelper &psH, size_t *P=NULL, size_t *Q=NULL)`
- `template<class Field >`  
`Field::Element_ptr Solve (const Field &F, const size_t M, typename Field::Element_ptr A, const size_t Ida, typename Field::Element_ptr x, const int incx, typename Field::ConstElement_ptr b, const int incb)`  
*Solves a linear system  $AX = b$  using PLUQ factorization.*
- `template<class Field , class PSHelper >`  
`Field::Element_ptr Solve (const Field &F, const size_t M, typename Field::Element_ptr A, const size_t Ida, typename Field::Element_ptr x, const int incx, typename Field::ConstElement_ptr b, const int incb, PSHelper &psH)`
- `template<class Field >`  
`Field::Element_ptr pSolve (const Field &F, const size_t M, typename Field::Element_ptr A, const size_t Ida, typename Field::Element_ptr x, const int incx, typename Field::ConstElement_ptr b, const int incb, size_t numthreads=0)`
- `template<class Field >`  
`*void RandomNullSpaceVector (const Field &F, const FFLAS::FFLAS_SIDE Side, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t Ida, typename Field::Element_ptr X, const size_t incX)`  
*Solve  $LX = B$  or  $XL = B$  in place.*
- `template<class Field >`  
`size_t NullSpaceBasis (const Field &F, const FFLAS::FFLAS_SIDE Side, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t Ida, typename Field::Element_ptr &NS, size_t &Idn, size_t &NSdim)`  
*Computes a basis of the Left/Right nullspace of the matrix A.*
- `template<class Field >`  
`size_t RowRankProfile (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t Ida, size_t *&rkprofile, const FFPACK_LU_TAG LuTag=FpackSlabRecursive)`  
*Computes the row rank profile of A.*
- `template<class Field >`  
`size_t pRowRankProfile (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t Ida, size_t *&rkprofile, size_t numthreads=0, const FFPACK_LU_TAG LuTag=FpackTileRecursive)`
- `template<class Field , class PSHelper >`  
`size_t RowRankProfile (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t Ida, size_t *&rkprofile, const FFPACK_LU_TAG LuTag, PSHelper &psH)`
- `template<class Field >`  
`size_t ColumnRankProfile (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t Ida, size_t *&rkprofile, const FFPACK_LU_TAG LuTag=FpackSlabRecursive)`  
*Computes the column rank profile of A.*
- `template<class Field >`  
`size_t pColumnRankProfile (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t Ida, size_t *&rkprofile, size_t numthreads=0, const FFPACK_LU_TAG LuTag=FpackTileRecursive)`

- `template<class Field, class PSHelper>`  
`size_t ColumnRankProfile (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *rkprofile, const FFPACK_LU_TAG LuTag, PSHelper &psH)`
- `void RankProfileFromLU (const size_t *P, const size_t N, const size_t R, size_t *rkprofile, const FFPACK_LU_TAG LuTag)`  
*Recovers the column/row rank profile from the permutation of an LU decomposition.*
- `size_t LeadingSubmatrixRankProfiles (const size_t M, const size_t N, const size_t R, const size_t LSm, const size_t LSn, const size_t *P, const size_t *Q, size_t *RRP, size_t *CRP)`  
*Recovers the row and column rank profiles of any leading submatrix from the PLUQ decomposition.*
- `template<class Field>`  
`size_t RowRankProfileSubmatrixIndices (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *rowindices, size_t *colindices, size_t &R)`  
*RowRankProfileSubmatrixIndices.*
- `template<class Field>`  
`size_t ColRankProfileSubmatrixIndices (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *rowindices, size_t *colindices, size_t &R)`  
*Computes the indices of the submatrix  $r \times r$   $X$  of  $A$  whose columns correspond to the column rank profile of  $A$ .*
- `template<class Field>`  
`size_t RowRankProfileSubmatrix (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr &X, size_t &R)`  
*Computes the  $r \times r$  submatrix  $X$  of  $A$ , by picking the row rank profile rows of  $A$ .*
- `template<class Field>`  
`size_t ColRankProfileSubmatrix (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr &X, size_t &R)`  
*Compute the  $r \times r$  submatrix  $X$  of  $A$ , by picking the row rank profile rows of  $A$ .*
- `template<class Field>`  
`void getTriangular (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG diag, const size_t M, const size_t N, const size_t R, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr T, const size_t ldt, const bool OnlyNonZeroVectors=false)`  
*Extracts a triangular matrix from a compact storage  $A=L\backslash U$  of rank  $R$ .*
- `template<class Field>`  
`void getTriangular (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG diag, const size_t M, const size_t N, const size_t R, typename Field::Element_ptr A, const size_t lda)`  
*Cleans up a compact storage  $A=L\backslash U$  to reveal a triangular matrix of rank  $R$ .*
- `template<class Field>`  
`void getEchelonForm (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG diag, const size_t M, const size_t N, const size_t R, const size_t *P, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr T, const size_t ldt, const bool OnlyNonZeroVectors=false, const FFPACK_LU_TAG LuTag=FpackSlabRecursive)`  
*Extracts a matrix in echelon form from a compact storage  $A=L\backslash U$  of rank  $R$  obtained by RowEchelonForm or ColumnEchelonForm.*
- `template<class Field>`  
`void getEchelonForm (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG diag, const size_t M, const size_t N, const size_t R, const size_t *P, typename Field::Element_ptr A, const size_t lda, const FFPACK_LU_TAG LuTag=FpackSlabRecursive)`  
*Cleans up a compact storage  $A=L\backslash U$  obtained by RowEchelonForm or ColumnEchelonForm to reveal an echelon form of rank  $R$ .*
- `template<class Field>`  
`void getEchelonTransform (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG diag, const size_t M, const size_t N, const size_t R, const size_t *P, const size_t *Q, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr T, const size_t ldt, const FFPACK_LU_TAG LuTag=FpackSlabRecursive)`  
*Extracts a transformation matrix to echelon form from a compact storage  $A=L\backslash U$  of rank  $R$  obtained by RowEchelonForm or ColumnEchelonForm.*



- `template<class Field >`  
`void getReducedEchelonForm (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const size_t M, const size_t N, const size_t R, const size_t *P, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr T, const size_t ldt, const bool OnlyNonZeroVectors=false, const FFPACK_LU_TAG LuTag=FFpackSlabRecursive)`  
*Extracts a matrix in echelon form from a compact storage  $A=L\backslash U$  of rank  $R$  obtained by `ReducedRowEchelonForm` or `ReducedColumnEchelonForm` with `transform = true`.*
- `template<class Field >`  
`void getReducedEchelonForm (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const size_t M, const size_t N, const size_t R, const size_t *P, typename Field::Element_ptr A, const size_t lda, const FFPACK_LU_TAG LuTag=FFpackSlabRecursive)`  
*Cleans up a compact storage  $A=L\backslash U$  of rank  $R$  obtained by `ReducedRowEchelonForm` or `ReducedColumnEchelonForm` with `transform = true`.*
- `template<class Field >`  
`void getReducedEchelonTransform (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const size_t M, const size_t N, const size_t R, const size_t *P, const size_t *Q, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr T, const size_t ldt, const FFPACK_LU_TAG LuTag=FFpackSlabRecursive)`  
*Extracts a transformation matrix to echelon form from a compact storage  $A=L\backslash U$  of rank  $R$  obtained by `RowEchelonForm` or `ColumnEchelonForm`.*
- `void PLUQtoEchelonPermutation (const size_t N, const size_t R, const size_t *P, size_t *outPerm)`  
*Auxiliary routine: determines the permutation that changes a PLUQ decomposition into a echelon form revealing PLUQ decomposition.*
- `template<class Field >`  
`size_t LTBruhatGen (const Field &Fi, const FFLAS::FFLAS_DIAG diag, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q)`  
*LTBruhatGen Suppose  $A$  is Left Triangular Matrix This procedure computes the Bruhat Representation of  $A$  and return the rank of  $A$ .*
- `template<class Field >`  
`void getLTBruhatGen (const Field &Fi, const size_t N, const size_t r, const size_t *P, const size_t *Q, typename Field::Element_ptr R, const size_t ldr)`  
*GetLTBruhatGen This procedure Computes the Rank Revealing Matrix based on the Bruhta representation of a Matrix.*
- `template<class Field >`  
`void getLTBruhatGen (const Field &Fi, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG diag, const size_t N, const size_t r, const size_t *P, const size_t *Q, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr T, const size_t ldt)`  
*GetLTBruhatGen This procedure computes the matrix  $L$  or  $U$  f the Bruhat Representation Suppose that  $A$  is the bruhat representation of a matrix.*
- `size_t LTQSorder (const size_t N, const size_t r, const size_t *P, const size_t *Q)`  
*LTQSorder This procedure computes the order of quasiseparability of a matrix.*
- `template<class Field >`  
`size_t CompressToBlockBiDiagonal (const Field &Fi, const FFLAS::FFLAS_UPLO Uplo, size_t N, size_t s, size_t r, const size_t *P, const size_t *Q, typename Field::Element_ptr A, size_t lda, typename Field::Element_ptr X, size_t ldx, size_t *K, size_t *M, size_t *T)`  
*CompressToBlockBiDiagonal This procedure compress a compact representation of a row echelon form or column echelon form.*
- `template<class Field >`  
`void ExpandBlockBiDiagonalToBruhat (const Field &Fi, const FFLAS::FFLAS_UPLO Uplo, size_t N, size_t s, size_t r, typename Field::Element_ptr A, size_t lda, typename Field::Element_ptr X, size_t ldx, size_t NbBlocks, size_t *K, size_t *M, size_t *T)`  
*ExpandBlockBiDiagonal This procedure expand a compact representation of a row echelon form or column echelon form.*
- `void Bruhat2EchelonPermutation (size_t N, size_t R, const size_t *P, const size_t *Q, size_t *M)`  
*Bruhat2EchelonPermutation ( $N, R, P, Q$ ) Compute  $M$  such that  $LM$  or  $MU$  is in echelon form where  $L$  or  $U$  are factors of the Bruhat Representation.*
- `size_t * Tinverter (size_t *T, size_t r)`



- `template<class Field >`  
`void ComputeRPermutation (const Field &Fi, size_t N, size_t r, const size_t *P, const size_t *Q, size_t *R, size_t *MU, size_t *ML)`
- `template<class Field >`  
`void productBruhatxTS (const Field &Fi, size_t N, size_t s, size_t r, const size_t *P, const size_t *Q, const typename Field::Element_ptr Xu, size_t ldu, size_t NbBlocksU, size_t *Ku, size_t *Tu, size_t *MU, const typename Field::Element_ptr XI, size_t ldl, size_t NbBlocksL, size_t *KI, size_t *TI, size_t *ML, typename Field::Element_ptr B, size_t t, size_t ldb, typename Field::Element_ptr C, size_t ldc)`  
*productBruhatxTS Compute the product between the CRE compact representation of a matrix A and B a tall matrix*
- `template<class Field >`  
`Field::Element_ptr LQUPTolInverseOfFullRankMinor (const Field &F, const size_t rank, typename Field::Element_ptr A_factors, const size_t lda, const size_t *QtPointer, typename Field::Element_ptr X, const size_t ldx)`  
*LQUPTolInverseOfFullRankMinor.*
- `template<class Field >`  
`void RandomNullSpaceVector (const Field &F, const FFLAS::FFLAS_SIDE Side, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr X, const size_t incX)`  
*Solve  $LX = B$  or  $XL = B$  in place.*
- `template<class Field >`  
`void solveLB (const Field &F, const FFLAS::FFLAS_SIDE Side, const size_t M, const size_t N, const size_t R, typename Field::Element_ptr L, const size_t ldl, const size_t *Q, typename Field::Element_ptr B, const size_t ldb)`
- `template<class Field >`  
`void solveLB2 (const Field &F, const FFLAS::FFLAS_SIDE Side, const size_t M, const size_t N, const size_t R, typename Field::Element_ptr L, const size_t ldl, const size_t *Q, typename Field::Element_ptr B, const size_t ldb)`
- `size_t * TInverter (const size_t *T, size_t r)`
- `template<class Field >`  
`void ComputeRPermutation (const Field &Fi, size_t N, size_t r, const size_t *P, const size_t *Q, size_t *R, const size_t *MU, const size_t *ML)`
- `template<class Field >`  
`Field::Element_ptr expandLCRE (const Field &Fi, size_t N, size_t s, size_t r, size_t *R, size_t i, typename Field::ConstElement_ptr Xu, size_t ldu, size_t NbBlocksU, const size_t *Ku, const size_t *Tuinv, typename Field::ConstElement_ptr XI, size_t ldl, size_t NbBlocksL, const size_t *KI, const size_t *Tlinv, typename Field::Element_ptr CRE, size_t ldcre)`  
*Expands an anti-diagonal block of a left triangular matrix from its compact Bruhat representation.*
- `template<class Field >`  
`void productBruhatxTS (const Field &Fi, size_t N, size_t s, size_t r, size_t t, const size_t *P, const size_t *Q, typename Field::ConstElement_ptr Xu, size_t ldu, size_t NbBlocksU, const size_t *Ku, const size_t *Tu, const size_t *MU, typename Field::ConstElement_ptr XI, size_t ldl, size_t NbBlocksL, const size_t *KI, const size_t *TI, const size_t *ML, typename Field::Element_ptr B, size_t ldb, const typename Field::Element beta, typename Field::Element_ptr D, size_t ldd)`  
*Compute the product of a left-triangular quasi-separable matrix A, represented by a compact Bruhat generator, with a dense rectangular matrix B:  $C \leftarrow A \times B + \text{beta}C$ .*
- `template<class Field, class Polynomial >`  
`std::list< Polynomial > & Danilevski (const Field &F, std::list< Polynomial > &charp, const size_t N, typename Field::Element_ptr A, const size_t lda)`
- `template<class Field >`  
`Field::Element_ptr buildMatrix (const Field &F, typename Field::ConstElement_ptr E, typename Field::ConstElement_ptr C, const size_t lda, const size_t *B, const size_t *T, const size_t me, const size_t mc, const size_t lambda, const size_t mu)`
- `FFPACK::RNSInteger< FFPACK::rns_double >::Element_ptr CharPoly (const FFPACK::RNSInteger< FFPACK::rns_double > &F, typename FFPACK::RNSInteger< FFPACK::rns_double >::Element_ptr charp, const size_t N, typename FFPACK::RNSInteger< FFPACK::rns_double >::Element_ptr A, const size_t lda, Givaro::ZRing< Givaro::Integer >::RandIter &G, const FFPACK_CHARPOLY_TAG CharpTag, size_t degree)`

- `template<> Givaro::Poly1Dom< Givaro::ZRing< Givaro::Integer > >::Element & CharPoly (const Givaro::Poly1Dom< Givaro::ZRing< Givaro::Integer > > &R, Givaro::Poly1Dom< Givaro::ZRing< Givaro::Integer > >::Element &charp, const size_t N, Givaro::Integer *A, const size_t Ida, Givaro::ZRing< Givaro::Integer >::RandIter &G, const FFPACK_CHARPOLY_TAG CharpTag, size_t degree)`
- `template<class PSHelper >  
FFPACK::RNSInteger< FFPACK::rns_double >::Element_ptr & Det (const FFPACK::RNSInteger< FFPACK::rns_double > &F, typename FFPACK::RNSInteger< FFPACK::rns_double >::Element_ptr &det, const size_t N, typename FFPACK::RNSInteger< FFPACK::rns_double >::Element_ptr A, const size_t Ida, const PSHelper &psH)`
- `template<class PSHelper >  
Givaro::Integer & Det (const Givaro::ZRing< Givaro::Integer > &F, Givaro::Integer &det, const size_t N, Givaro::Integer *A, const size_t Ida, const PSHelper &psH, size_t *P, size_t *Q)`
- `template<class Field >  
bool fsytrf_BC_Crout (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t Ida, typename Field::Element_ptr Dinv, const size_t incDinv)`
- `template<class Field >  
size_t fsytrf_BC_RL (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t Ida, typename Field::Element_ptr Dinv, const size_t incDinv)`
- `template<class Field >  
size_t fsytrf_UP_RPM_BC_RL (const Field &F, const size_t N, typename Field::Element_ptr A, const size_t Ida, typename Field::Element_ptr Dinv, const size_t incDinv, size_t *P)`
- `template<class Field >  
size_t fsytrf_LOW_RPM_BC_Crout (const Field &F, const size_t N, typename Field::Element_ptr A, const size_t Ida, typename Field::Element_ptr Dinv, const size_t incDinv, size_t *P)`
- `template<class Field >  
size_t fsytrf_UP_RPM_BC_Crout (const Field &F, const size_t N, typename Field::Element_ptr A, const size_t Ida, typename Field::Element_ptr Dinv, const size_t incDinv, size_t *P)`
- `template<class Field >  
size_t fsytrf_UP_RPM (const Field &F, const size_t N, typename Field::Element_ptr A, const size_t Ida, typename Field::Element_ptr Dinv, const size_t incDinv, size_t *P, size_t BCThreshold)`
- `template<class Field >  
bool fsytrf_nonunit (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t Ida, typename Field::Element_ptr Dinv, const size_t incDinv, FFLAS::ParSeqHelper::Sequential seq, size_t threshold)`
- `template<class Field, class Cut, class Param >  
bool fsytrf_nonunit (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t Ida, typename Field::Element_ptr Dinv, const size_t incDinv, FFLAS::ParSeqHelper::Parallel< Cut, Param > par, size_t threshold)`
- `template<class Field >  
size_t fsytrf_RPM (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t Ida, size_t *P, size_t threshold)`
- `template<class Field >  
void getTridiagonal (const Field &F, const size_t N, const size_t R, typename Field::ConstElement_ptr A, const size_t Ida, size_t *P, typename Field::Element_ptr T, const size_t ldt)`
- `template<class Field >  
size_t LUdivine_gauss (const Field &F, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t Ida, size_t *P, size_t *Q, const FFPACK::FFPACK_LU_TAG LuTag)`
- `template<class Field >  
size_t LUdivine_small (const Field &F, const FFLAS::FFLAS_DIAG Diag, const FFLAS::FFLAS_TRANSPOSE trans, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t Ida, size_t *P, size_t *Q, const FFPACK::FFPACK_LU_TAG LuTag)`
- `template<class Field >  
size_t LUdivine (const Field &F, const FFLAS::FFLAS_DIAG Diag, const FFLAS::FFLAS_TRANSPOSE trans, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t Ida, size_t *P, size_t *Q, const FFPACK::FFPACK_LU_TAG LuTag, const size_t cutoff)`

- `template<> size_t LUdivine (const Givaro::Modular< Givaro::Integer > &F, const FFLAS::FFLAS_DIAG Diag, const FFLAS::FFLAS_TRANSPOSE trans, const size_t M, const size_t N, typename Givaro::Integer *A, const size_t lda, size_t *P, size_t *Q, const FFPACK::FFPACK_LU_TAG LuTag, const size_t cutoff)`
- `template<class Field >`  
`void MonotonicCompress (const Field &F, const FFLAS::FFLAS_SIDE Side, const size_t M, typename Field::Element_ptr A, const size_t lda, const size_t incA, const size_t *MathP, const size_t R, const size_t maxpiv, const size_t rowstomove, const std::vector< bool > &ispiv)`
- `template<class Field >`  
`void MonotonicCompressMorePivots (const Field &F, const FFLAS::FFLAS_SIDE Side, const size_t M, typename Field::Element_ptr A, const size_t lda, const size_t incA, const size_t *MathP, const size_t R, const size_t rowstomove, const size_t lenP)`
- `template<class Field >`  
`void MonotonicCompressCycles (const Field &F, const FFLAS::FFLAS_SIDE Side, const size_t M, typename Field::Element_ptr A, const size_t lda, const size_t incA, const size_t *MathP, const size_t lenP)`
- `template<class Field >`  
`void MonotonicExpand (const Field &F, const FFLAS::FFLAS_SIDE Side, const size_t M, typename Field::Element_ptr A, const size_t lda, const size_t incA, const size_t *MathP, const size_t R, const size_t maxpiv, const size_t rowstomove, const std::vector< bool > &ispiv)`
- `template<class Field >`  
`void applyP_block (const Field &F, const FFLAS::FFLAS_SIDE Side, const FFLAS::FFLAS_TRANSPOSE Trans, const size_t M, const size_t ibeg, const size_t iend, typename Field::Element_ptr A, const size_t lda, const size_t *P)`
- `template<class Field >`  
`void doApplyS (const Field &F, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr tmp, const size_t width, const size_t M2, const size_t R1, const size_t R2, const size_t R3, const size_t R4)`
- `template<class Field >`  
`void MatrixApplyS (const Field &F, typename Field::Element_ptr A, const size_t lda, const size_t width, const size_t M2, const size_t R1, const size_t R2, const size_t R3, const size_t R4)`
- `template<class Field >`  
`void MatrixApplyS (const Field &F, typename Field::Element_ptr A, const size_t lda, const size_t width, const size_t M2, const size_t R1, const size_t R2, const size_t R3, const size_t R4, const FFLAS::ParSeqHelper::Sequential seq)`
- `template<class Field, class Cut, class Param >`  
`void MatrixApplyS (const Field &F, typename Field::Element_ptr A, const size_t lda, const size_t width, const size_t M2, const size_t R1, const size_t R2, const size_t R3, const size_t R4, const FFLAS::ParSeqHelper::Parallel< Cut, Param > par)`
- `template<class T >`  
`void PermApplyS (T *A, const size_t lda, const size_t width, const size_t M2, const size_t R1, const size_t R2, const size_t R3, const size_t R4)`
- `template<class Field >`  
`void doApplyT (const Field &F, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr tmp, const size_t width, const size_t N2, const size_t R1, const size_t R2, const size_t R3, const size_t R4)`
- `template<class Field >`  
`void MatrixApplyT (const Field &F, typename Field::Element_ptr A, const size_t lda, const size_t width, const size_t N2, const size_t R1, const size_t R2, const size_t R3, const size_t R4)`
- `template<class Field >`  
`void MatrixApplyT (const Field &F, typename Field::Element_ptr A, const size_t lda, const size_t width, const size_t N2, const size_t R1, const size_t R2, const size_t R3, const size_t R4, const FFLAS::ParSeqHelper::Sequential seq)`
- `template<class Field, class Cut, class Param >`  
`void MatrixApplyT (const Field &F, typename Field::Element_ptr A, const size_t lda, const size_t width, const size_t N2, const size_t R1, const size_t R2, const size_t R3, const size_t R4, const FFLAS::ParSeqHelper::Parallel< Cut, Param > par)`
- `template<class T >`  
`void PermApplyT (T *A, const size_t lda, const size_t width, const size_t N2, const size_t R1, const size_t R2, const size_t R3, const size_t R4)`
- `void composePermutationsLLL (size_t *P1, const size_t *P2, const size_t R, const size_t N)`

*Computes  $P1 \times \text{Diag}(I\_R, P2)$  where  $P1$  is a LAPACK and  $P2$  a LAPACK permutation and store the result in  $P1$  as a LAPACK permutation.*

- `void composePermutationsLLM (size_t *MathP, const size_t *P1, const size_t *P2, const size_t R, const size_t N)`

*Computes  $P1 \times \text{Diag}(I\_R, P2)$  where  $P1$  is a LAPACK and  $P2$  a LAPACK permutation and store the result in  $MathP$  as a  $MathPermutation$  format.*

- `void composePermutationsMLM (size_t *MathP1, const size_t *P2, const size_t R, const size_t N)`

*Computes  $MathP1 \times \text{Diag}(I\_R, P2)$  where  $MathP1$  is a  $MathPermutation$  and  $P2$  a LAPACK permutation and store the result in  $MathP1$  as a  $MathPermutation$  format.*

- `void cyclic_shift_mathPerm (size_t *P, const size_t s)`
- `template<class Field >`  
`void cyclic_shift_row_col (const Field &F, typename Field::Element_ptr A, size_t m, size_t n, size_t lda)`
- `template<class Field >`  
`void cyclic_shift_row (const Field &F, typename Field::Element_ptr A, size_t m, size_t n, size_t lda)`
- `template<typename T >`  
`void cyclic_shift_row (const RNSIntegerMod< T > &F, typename T::Element_ptr A, size_t m, size_t n, size_t lda)`
- `template<class Field >`  
`void cyclic_shift_col (const Field &F, typename Field::Element_ptr A, size_t m, size_t n, size_t lda)`
- `template<typename T >`  
`void cyclic_shift_col (const RNSIntegerMod< T > &F, typename T::Element_ptr A, size_t m, size_t n, size_t lda)`
- `template<class Field >`  
`size_t PLUQ_basecaseV3 (const Field &Fi, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::Element *A, const size_t lda, size_t *P, size_t *Q)`
- `template<class Field >`  
`size_t PLUQ_basecaseV2 (const Field &Fi, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::Element *A, const size_t lda, size_t *P, size_t *Q)`
- `template<class Field >`  
`size_t PLUQ_basecaseCrout (const Field &Fi, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q)`
- `template<class Field >`  
`size_t _PLUQ (const Field &Fi, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q, size_t BCThreshold)`
- `template<class Cut, class Param >`  
`size_t PLUQ (const Givaro::Modular< Givaro::Integer > &F, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Givaro::Integer *A, const size_t lda, size_t *P, size_t *Q, size_t BCThreshold, FFLAS::ParSeqHelper::Parallel< Cut, Param > &PSHelper)`
- `template<class Field >`  
`void threads_fgemm (const size_t m, const size_t n, const size_t r, int nbthreads, size_t *W1, size_t *W2, size_t *W3, size_t gamma)`
- `template<class Field >`  
`void threads_ftrsm (const size_t m, const size_t n, int nbthreads, size_t *t1, size_t *t2)`
- `template<class Field >`  
`size_t PLUQ (const Field &Fi, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q, const FFLAS::ParSeqHelper::Parallel< FFLAS::CuttingStrategy::Recursive, FFLAS::StrategyParameter::Threads > &PSHelper)`
- `template<> rns_double_elt_ptr fflas_const_cast (rns_double_elt_cstptr x)`
- `template<> rns_double_elt_cstptr fflas_const_cast (rns_double_elt_ptr x)`
- `template<typename Base_t >`  
`void cyclic_shift_row_col (Base_t *A, size_t m, size_t n, size_t lda)`
- `template INST_OR_DECL void cyclic_shift_row (const FFLAS_FIELD< FFLAS_ELT > &F, FFLAS_ELT *A, size_t m, size_t n, size_t lda)`
- `template INST_OR_DECL void cyclic_shift_col (const FFLAS_FIELD< FFLAS_ELT > &F, FFLAS_ELT *A, size_t m, size_t n, size_t lda)`

- `template INST_OR_DECL void applyP (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS::FFLAS_SIDE Side, const FFLAS::FFLAS_TRANSPOSE Trans, const size_t M, const size_t ibeg, const size_t iend, FFLAS_ELT *A, const size_t lda, const size_t *P)`
- `template INST_OR_DECL void fgetrs (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS::FFLAS_SIDE Side, const size_t M, const size_t N, const size_t R, FFLAS_ELT *A, const size_t lda, const size_t *P, const size_t *Q, FFLAS_ELT *B, const size_t ldb, int *info)`
- `template INST_OR_DECL FFLAS_ELT * fgetrs (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS::FFLAS_SIDE Side, const size_t M, const size_t N, const size_t NRHS, const size_t R, FFLAS_ELT *A, const size_t lda, const size_t *P, const size_t *Q, FFLAS_ELT *X, const size_t idx, const FFLAS_ELT *B, const size_t ldb, int *info)`
- `template INST_OR_DECL size_t fgesv (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS::FFLAS_SIDE Side, const size_t M, const size_t N, FFLAS_ELT *A, const size_t lda, FFLAS_ELT *B, const size_t ldb, int *info)`
- `template INST_OR_DECL size_t fgesv (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS::FFLAS_SIDE Side, const size_t M, const size_t N, const size_t NRHS, FFLAS_ELT *A, const size_t lda, FFLAS_ELT *X, const size_t idx, const FFLAS_ELT *B, const size_t ldb, int *info)`
- `template INST_OR_DECL void ftrtri (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG Diag, const size_t N, FFLAS_ELT *A, const size_t lda, const size_t threshold)`
- `template INST_OR_DECL void trinv_left (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t N, const FFLAS_ELT *L, const size_t ldl, FFLAS_ELT *X, const size_t idx)`
- `template INST_OR_DECL void ftrtrm (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS::FFLAS_SIDE side, const FFLAS::FFLAS_DIAG diag, const size_t N, FFLAS_ELT *A, const size_t lda)`
- `template INST_OR_DECL size_t PLUQ (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, FFLAS_ELT *A, const size_t lda, size_t *P, size_t *Q)`
- `template INST_OR_DECL size_t LUdivine (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS::FFLAS_DIAG Diag, const FFLAS::FFLAS_TRANSPOSE trans, const size_t M, const size_t N, FFLAS_ELT *A, const size_t lda, size_t *P, size_t *Qt, const FFPACK_LU_TAG LuTag, const size_t cutoff)`
- `template INST_OR_DECL size_t LUdivine_small (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS::FFLAS_DIAG Diag, const FFLAS::FFLAS_TRANSPOSE trans, const size_t M, const size_t N, FFLAS_ELT *A, const size_t lda, size_t *P, size_t *Q, const FFPACK_LU_TAG LuTag)`
- `template INST_OR_DECL size_t LUdivine_gauss (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, FFLAS_ELT *A, const size_t lda, size_t *P, size_t *Q, const FFPACK_LU_TAG LuTag)`
- `template INST_OR_DECL size_t RowEchelonForm (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t M, const size_t N, FFLAS_ELT *A, const size_t lda, size_t *P, size_t *Qt, const bool transform, const FFPACK_LU_TAG LuTag)`
- `template INST_OR_DECL size_t ReducedRowEchelonForm (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t M, const size_t N, FFLAS_ELT *A, const size_t lda, size_t *P, size_t *Qt, const bool transform, const FFPACK_LU_TAG LuTag)`
- `template INST_OR_DECL size_t ColumnEchelonForm (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t M, const size_t N, FFLAS_ELT *A, const size_t lda, size_t *P, size_t *Qt, const bool transform, const FFPACK_LU_TAG LuTag)`
- `template INST_OR_DECL size_t ReducedColumnEchelonForm (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t M, const size_t N, FFLAS_ELT *A, const size_t lda, size_t *P, size_t *Qt, const bool transform, const FFPACK_LU_TAG LuTag)`
- `template INST_OR_DECL FFLAS_ELT * Invert (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t M, FFLAS_ELT *A, const size_t lda, int &>nullity)`
- `template INST_OR_DECL FFLAS_ELT * Invert (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t M, const FFLAS_ELT *A, const size_t lda, FFLAS_ELT *X, const size_t idx, int &>nullity)`
- `template INST_OR_DECL FFLAS_ELT * Invert2 (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t M, FFLAS_ELT *A, const size_t lda, FFLAS_ELT *X, const size_t idx, int &>nullity)`
- `template INST_OR_DECL std::list< Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > >::Element > &CharPoly (const Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > > &R, std::list< Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > >::Element > &charp, const size_t N, FFLAS_ELT *A, const size_t lda, FFLAS_FIELD< FFLAS_ELT >::RandIter &G, const FFPACK_CHARPOLY_TAG CharpTag, const size_t degree)`

- `template INST_OR_DECL Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > >::Element & CharPoly (const Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > > &R, Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > >::Element &charp, const size_t N, FFLAS_ELT *A, const size_t lda, FFLAS_FIELD< FFLAS_ELT >::RandIter &G, const FFPACK_CHARPOLY_TAG CharpTag, const size_t degree)`
- `template INST_OR_DECL Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > >::Element & CharPoly (const Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > > &R, Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > >::Element &charp, const size_t N, FFLAS_ELT *A, const size_t lda, const FFPACK_CHARPOLY_TAG CharpTag, const size_t degree)`
- `template INST_OR_DECL std::vector< FFLAS_ELT > & MinPoly (const FFLAS_FIELD< FFLAS_ELT > &F, std::vector< FFLAS_ELT > &minP, const size_t N, const FFLAS_ELT *A, const size_t lda, FFLAS_FIELD< FFLAS_ELT >::RandIter &G)`
- `template INST_OR_DECL std::vector< FFLAS_ELT > & MinPoly (const FFLAS_FIELD< FFLAS_ELT > &F, std::vector< FFLAS_ELT > &minP, const size_t N, const FFLAS_ELT *A, const size_t lda)`
- `template INST_OR_DECL std::vector< FFLAS_ELT > & MatVecMinPoly (const FFLAS_FIELD< FFLAS_ELT > &F, std::vector< FFLAS_ELT > &minP, const size_t N, const FFLAS_ELT *A, const size_t lda, const FFLAS_ELT *V, const size_t incv)`
- `template INST_OR_DECL size_t KrylovElim (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t M, const size_t N, FFLAS_ELT *A, const size_t lda, size_t *P, size_t *Q, const size_t deg, size_t *iterates, size_t *inviterates, const size_t maxit, size_t virt)`
- `template INST_OR_DECL size_t SpecRankProfile (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t M, const size_t N, FFLAS_ELT *A, const size_t lda, const size_t deg, size_t *rankProfile)`
- `template INST_OR_DECL size_t Rank (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t M, const size_t N, FFLAS_ELT *A, const size_t lda)`
- `template INST_OR_DECL bool IsSingular (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t M, const size_t N, FFLAS_ELT *A, const size_t lda)`
- `template INST_OR_DECL FFLAS_ELT & Det (const FFLAS_FIELD< FFLAS_ELT > &F, FFLAS_ELT &det, const size_t N, FFLAS_ELT *A, const size_t lda, size_t *P, size_t *Q)`
- `template INST_OR_DECL FFLAS_ELT & Det (const FFLAS_FIELD< FFLAS_ELT > &F, FFLAS_ELT &det, const size_t N, FFLAS_ELT *A, const size_t lda, const FFLAS::ParSeqHelper::Parallel< FFLAS::CuttingStrategy::Recursive, FFLAS::StrategyParameter::Threads > &parH, size_t *P, size_t *Q)`
- `template INST_OR_DECL FFLAS_ELT * Solve (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t M, FFLAS_ELT *A, const size_t lda, FFLAS_ELT *x, const int incx, const FFLAS_ELT *b, const int incb)`
- `template INST_OR_DECL void solveLB (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS::FFLAS_SIDE Side, const size_t M, const size_t N, const size_t R, FFLAS_ELT *L, const size_t ldl, const size_t *Q, FFLAS_ELT *B, const size_t ldb)`
- `template INST_OR_DECL void solveLB2 (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS::FFLAS_SIDE Side, const size_t M, const size_t N, const size_t R, FFLAS_ELT *L, const size_t ldl, const size_t *Q, FFLAS_ELT *B, const size_t ldb)`
- `template INST_OR_DECL void RandomNullSpaceVector (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS::FFLAS_SIDE Side, const size_t M, const size_t N, FFLAS_ELT *A, const size_t lda, FFLAS_ELT *X, const size_t incX)`
- `template INST_OR_DECL size_t NullSpaceBasis (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS::FFLAS_SIDE Side, const size_t M, const size_t N, FFLAS_ELT *A, const size_t lda, FFLAS_ELT *NS, size_t &ldn, size_t &NSdim)`
- `template INST_OR_DECL size_t RowRankProfile (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t M, const size_t N, FFLAS_ELT *A, const size_t lda, size_t *rkprofile, const FFPACK_LU_TAG LuTag)`
- `template INST_OR_DECL size_t ColumnRankProfile (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t M, const size_t N, FFLAS_ELT *A, const size_t lda, size_t *rkprofile, const FFPACK_LU_TAG LuTag)`
- `template INST_OR_DECL size_t RowRankProfileSubmatrixIndices (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t M, const size_t N, FFLAS_ELT *A, const size_t lda, size_t *rowindices, size_t *colindices, size_t &R)`
- `template INST_OR_DECL size_t ColRankProfileSubmatrixIndices (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t M, const size_t N, FFLAS_ELT *A, const size_t lda, size_t *rowindices, size_t *colindices, size_t &R)`
- `template INST_OR_DECL size_t RowRankProfileSubmatrix (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t M, const size_t N, FFLAS_ELT *A, const size_t lda, FFLAS_ELT *X, size_t &R)`



- `template INST_OR_DECL size_t ColRankProfileSubmatrix (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t M, const size_t N, FFLAS_ELT *A, const size_t lda, FFLAS_ELT *&X, size_t &R)`
- `template INST_OR_DECL void getTriangular< FFLAS_FIELD< FFLAS_ELT > > (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG diag, const size_t M, const size_t N, const size_t R, const FFLAS_ELT *A, const size_t lda, FFLAS_ELT *T, const size_t ldt, const bool OnlyNonZeroVectors)`
- `template INST_OR_DECL void getTriangular< FFLAS_FIELD< FFLAS_ELT > > (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG diag, const size_t M, const size_t N, const size_t R, FFLAS_ELT *A, const size_t lda)`
- `template INST_OR_DECL void getEchelonForm< FFLAS_FIELD< FFLAS_ELT > > (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG diag, const size_t M, const size_t N, const size_t R, const size_t *P, const FFLAS_ELT *A, const size_t lda, FFLAS_ELT *T, const size_t ldt, const bool OnlyNonZeroVectors, const FFPACK_LU_TAG LuTag)`
- `template INST_OR_DECL void getEchelonForm< FFLAS_FIELD< FFLAS_ELT > > (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG diag, const size_t M, const size_t N, const size_t R, const size_t *P, FFLAS_ELT *A, const size_t lda, const FFPACK_LU_TAG LuTag)`
- `template INST_OR_DECL void getEchelonTransform< FFLAS_FIELD< FFLAS_ELT > > (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG diag, const size_t M, const size_t N, const size_t R, const size_t *P, const size_t *Q, const FFLAS_ELT *A, const size_t lda, FFLAS_ELT *T, const size_t ldt, const FFPACK_LU_TAG LuTag)`
- `template INST_OR_DECL void getReducedEchelonForm< FFLAS_FIELD< FFLAS_ELT > > (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS::FFLAS_UPLO Uplo, const size_t M, const size_t N, const size_t R, const size_t *P, const FFLAS_ELT *A, const size_t lda, FFLAS_ELT *T, const size_t ldt, const bool OnlyNonZeroVectors, const FFPACK_LU_TAG LuTag)`
- `template INST_OR_DECL void getReducedEchelonForm< FFLAS_FIELD< FFLAS_ELT > > (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS::FFLAS_UPLO Uplo, const size_t M, const size_t N, const size_t R, const size_t *P, FFLAS_ELT *A, const size_t lda, const FFPACK_LU_TAG LuTag)`
- `template INST_OR_DECL void getReducedEchelonTransform< FFLAS_FIELD< FFLAS_ELT > > (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS::FFLAS_UPLO Uplo, const size_t M, const size_t N, const size_t R, const size_t *P, const size_t *Q, const FFLAS_ELT *A, const size_t lda, FFLAS_ELT *T, const size_t ldt, const FFPACK_LU_TAG LuTag)`
- `template INST_OR_DECL FFLAS_ELT * LQUPtoInverseOfFullRankMinor (const FFLAS_FIELD< FFLAS_ELT > &F, const size_t rank, FFLAS_ELT *A_factors, const size_t lda, const size_t *QtPointer, FFLAS_ELT *X, const size_t ldx)`
- `template<class T, class CT = const T>  
T fflas_const_cast (CT x)`
- `Failure & failure ()`
- `template<class T >  
bool isOdd (const T &a)`
- `bool isOdd (const float &a)`
- `bool isOdd (const double &a)`
- `template<class Field, class RandIter >  
Field::Element_ptr NonZeroRandomMatrix (const Field &F, size_t m, size_t n, typename Field::Element_ptr A, size_t lda, RandIter &G)  
Random non-zero Matrix.`
- `template<class Field, class RandIter >  
Field::Element_ptr NonZeroRandomMatrix (const Field &F, size_t m, size_t n, typename Field::Element_ptr A, size_t lda)  
Random non-zero Matrix.`
- `template<class Field, class RandIter >  
Field::Element_ptr RandomMatrix (const Field &F, size_t m, size_t n, typename Field::Element_ptr A, size_t lda, RandIter &G)  
Random Matrix.`
- `template<class Field >  
Field::Element_ptr RandomMatrix (const Field &F, size_t m, size_t n, typename Field::Element_ptr A, size_t lda)`

*Random Matrix.*

- `template<class Field , class RandIter >`  
`Field::Element_ptr RandomTriangularMatrix (const Field &F, size_t m, size_t n, const FFLAS::FFLAS_UPLO UpLo, const FFLAS::FFLAS_DIAG Diag, bool nonsingular, typename Field::Element_ptr A, size_t lda, RandIter &G)`

*Random Triangular Matrix.*

- `template<class Field >`  
`Field::Element_ptr RandomTriangularMatrix (const Field &F, size_t m, size_t n, const FFLAS::FFLAS_UPLO UpLo, const FFLAS::FFLAS_DIAG Diag, bool nonsingular, typename Field::Element_ptr A, size_t lda)`

*Random Triangular Matrix.*

- `size_t RandInt (size_t a, size_t b)`
- `template<class Field , class RandIter >`  
`Field::Element_ptr RandomSymmetricMatrix (const Field &F, size_t n, bool nonsingular, typename Field::Element_ptr A, size_t lda, RandIter &G)`

*Random Symmetric Matrix.*

- `template<class Field , class RandIter >`  
`Field::Element_ptr RandomMatrixWithRank (const Field &F, size_t m, size_t n, size_t r, typename Field::Element_ptr A, size_t lda, RandIter &G)`

*Random Matrix with prescribed rank.*

- `template<class Field >`  
`Field::Element_ptr RandomMatrixWithRank (const Field &F, size_t m, size_t n, size_t r, typename Field::Element_ptr A, size_t lda)`

*Random Matrix with prescribed rank.*

- `size_t * RandomIndexSubset (size_t N, size_t R, size_t *P)`  
*Pick uniformly at random a sequence of  $R$  distinct elements from the set  $\{0, \dots, N - 1\}$  using Knuth's shuffle.*
- `size_t * RandomPermutation (size_t N, size_t *P)`  
*Pick uniformly at random a permutation of size  $N$  stored in LAPACK format using Knuth's shuffle.*
- `void RandomRankProfileMatrix (size_t M, size_t N, size_t R, size_t *rows, size_t *cols)`  
*Pick uniformly at random an  $R$ -subpermutation of dimension  $M \times N$  : a matrix with only  $R$  non-zeros equal to one, in a random rook placement.*
- `void swapval (size_t k, size_t N, size_t *P, size_t val)`
- `void RandomSymmetricRankProfileMatrix (size_t N, size_t R, size_t *rows, size_t *cols)`  
*Pick uniformly at random a symmetric  $R$ -subpermutation of dimension  $N \times N$  : a symmetric matrix with only  $R$  non-zeros, all equal to one, in a random rook placement.*
- `void RandomLTQSRankProfileMatrix (size_t n, size_t r, size_t t, size_t *rows, size_t *cols)`
- `template<class Field , class RandIter >`  
`Field::Element_ptr RandomMatrixWithRankandRPM (const Field &F, size_t M, size_t N, size_t R, typename Field::Element_ptr A, size_t lda, const size_t *RRP, const size_t *CRP, RandIter &G)`

*Random Matrix with prescribed rank and rank profile matrix Creates an  $m \times n$  matrix with random entries and rank  $r$ .*

- `template<class Field >`  
`Field::Element_ptr RandomMatrixWithRankandRPM (const Field &F, size_t M, size_t N, size_t R, typename Field::Element_ptr A, size_t lda, const size_t *RRP, const size_t *CRP)`

*Random Matrix with prescribed rank and rank profile matrix Creates an  $m \times n$  matrix with random entries and rank  $r$ .*

- `template<class Field , class RandIter >`  
`Field::Element_ptr RandomSymmetricMatrixWithRankandRPM (const Field &F, size_t N, size_t R, typename Field::Element_ptr A, size_t lda, const size_t *RRP, const size_t *CRP, RandIter &G)`

*Random Symmetric Matrix with prescribed rank and rank profile matrix Creates an  $n \times n$  symmetric matrix with random entries and rank  $r$ .*

- `template<class Field >`  
`Field::Element_ptr RandomSymmetricMatrixWithRankandRPM (const Field &F, size_t M, size_t N, size_t R, typename Field::Element_ptr A, size_t lda, const size_t *RRP, const size_t *CRP)`

*Random Symmetric Matrix with prescribed rank and rank profile matrix Creates an  $n \times n$  symmetric matrix with random entries and rank  $r$ .*



- `template<class Field , class RandIter >`  
`Field::Element_ptr RandomMatrixWithRankandRandomRPM (const Field &F, size_t M, size_t N, size_t R, typename Field::Element_ptr A, size_t lda, RandIter &G)`  
*Random Matrix with prescribed rank, with random rank profile matrix Creates an  $m \times n$  matrix with random entries, rank  $r$  and with a rank profile matrix chosen uniformly at random.*
- `template<class Field >`  
`Field::Element_ptr RandomMatrixWithRankandRandomRPM (const Field &F, size_t M, size_t N, size_t R, typename Field::Element_ptr A, size_t lda)`  
*Random Matrix with prescribed rank, with random rank profile matrix Creates an  $m \times n$  matrix with random entries, rank  $r$  and with a rank profile matrix chosen uniformly at random.*
- `template<class Field , class RandIter >`  
`Field::Element_ptr RandomSymmetricMatrixWithRankandRandomRPM (const Field &F, size_t N, size_t R, typename Field::Element_ptr A, size_t lda, RandIter &G)`  
*Random Symmetric Matrix with prescribed rank, with random rank profile matrix Creates an  $n \times n$  matrix with random entries, rank  $r$  and with a rank profile matrix chosen uniformly at random.*
- `template<class Field >`  
`Field::Element_ptr RandomSymmetricMatrixWithRankandRandomRPM (const Field &F, size_t N, size_t R, typename Field::Element_ptr A, size_t lda)`  
*Random Symmetric Matrix with prescribed rank, with random rank profile matrix Creates an  $n \times n$  matrix with random entries, rank  $r$  and with a rank profile matrix chosen uniformly at random.*
- `template<class Field >`  
`Field::Element_ptr RandomMatrixWithDet (const Field &F, size_t n, const typename Field::Element d, typename Field::Element_ptr A, size_t lda)`  
*Random Matrix with prescribed det.*
- `template<class Field , class RandIter >`  
`Field::Element_ptr RandomMatrixWithDet (const Field &F, size_t n, const typename Field::Element d, typename Field::Element_ptr A, size_t lda, RandIter &G)`  
*Random Matrix with prescribed det.*
- `template<class Field , class RandIter >`  
`Field::Element_ptr RandomLTQSMMatrixWithRankandQSorder (Field &F, size_t n, size_t r, size_t t, typename Field::Element_ptr A, size_t lda, RandIter &G)`
- `template<typename Field >`  
`Field * chooseField (Givaro::Integer q, uint64_t b, uint64_t seed)`
- `template<> Givaro::ZRing< int32_t > * chooseField< Givaro::ZRing< int32_t > > (Givaro::Integer q, uint64_t b, uint64_t seed)`
- `template<> Givaro::ZRing< int64_t > * chooseField< Givaro::ZRing< int64_t > > (Givaro::Integer q, uint64_t b, uint64_t seed)`
- `template<> Givaro::ZRing< float > * chooseField< Givaro::ZRing< float > > (Givaro::Integer q, uint64_t b, uint64_t seed)`
- `template<> Givaro::ZRing< double > * chooseField< Givaro::ZRing< double > > (Givaro::Integer q, uint64_t b, uint64_t seed)`

### 11.21.1 Detailed Description

**Finite Field PACK** Set of elimination based routines for dense linear algebra.

This namespace enlarges the set of BLAS routines of the class **FFLAS**, with higher level routines based on elimination.

### 11.21.2 Typedef Documentation

#### 11.21.2.1 Checker\_PLUQ

```
template<class Field >
using Checker_PLUQ = FFLAS::Checker_Empty<Field>
```

**11.21.2.2 Checker\_Det**

```
template<class Field >
using Checker_Det = FFLAS::Checker_Empty<Field>
```

**11.21.2.3 Checker\_invert**

```
template<class Field >
using Checker_invert = FFLAS::Checker_Empty<Field>
```

**11.21.2.4 Checker\_charpoly**

```
template<class Field , class Polynomial >
using Checker_charpoly = FFLAS::Checker_Empty<Field>
```

**11.21.2.5 ForceCheck\_PLUQ**

```
template<class Field >
using ForceCheck_PLUQ = CheckerImplem_PLUQ<Field>
```

**11.21.2.6 ForceCheck\_Det**

```
template<class Field >
using ForceCheck_Det = CheckerImplem_Det<Field>
```

**11.21.2.7 ForceCheck\_invert**

```
template<class Field >
using ForceCheck_invert = CheckerImplem_invert<Field>
```

**11.21.2.8 ForceCheck\_charpoly**

```
template<class Field , class Polynomial >
using ForceCheck_charpoly = CheckerImplem_charpoly<Field,Polynomial>
```

**11.21.3 Function Documentation****11.21.3.1 LAPACKPerm2MathPerm()**

```
void LAPACKPerm2MathPerm (
 size_t * MathP,
 const size_t * LapackP,
 const size_t N) [inline]
```

Conversion of a permutation from LAPACK format to Math format.

### 11.21.3.2 MathPerm2LAPACKPerm()

```
void MathPerm2LAPACKPerm (
 size_t * LapackP,
 const size_t * MathP,
 const size_t N) [inline]
```

Conversion of a permutation from Maths format to LAPACK format.

### 11.21.3.3 applyP() [1/4]

```
template<class Field >
void applyP (
 const Field & F,
 const FFLAS::FFLAS_SIDE Side,
 const FFLAS::FFLAS_TRANSPOSE Trans,
 const size_t M,
 const size_t ibeg,
 const size_t iend,
 typename Field::Element_ptr A,
 const size_t lda,
 const size_t * P) [inline]
```

Computes  $P1 \times \text{Diag}(I_R, P2)$  where  $P1$  is a LAPACK and  $P2$  a LAPACK permutation and store the result in  $P1$  as a LAPACK permutation.

#### Parameters

|         |      |                                  |
|---------|------|----------------------------------|
| in, out | $P1$ | a LAPACK permutation of size N   |
|         | $P2$ | a LAPACK permutation of size N-R |

Applies a permutation  $P$  to the matrix  $A$ . Apply a permutation  $P$ , stored in the LAPACK format (a sequence of transpositions) between indices  $ibeg$  and  $iend$  of  $P$  to  $(iend-ibeg)$  vectors of size  $M$  stored in  $A$  (as column for NoTrans and rows for Trans).  $Side == \text{FFLAS::FflasLeft}$  for row permutation  $Side == \text{FFLAS::FflasRight}$  for a column permutation  $Trans == \text{FFLAS::FflasTrans}$  for the inverse permutation of  $P$

#### Parameters

|         |                                                                                                 |
|---------|-------------------------------------------------------------------------------------------------|
| $F$     | base field                                                                                      |
| $Side$  | decides if rows (FflasLeft) or columns (FflasRight) are permuted                                |
| $Trans$ | decides if the matrix is seen as columns (FflasTrans) or rows (FflasNoTrans)                    |
| $M$     | size of the elements to permute                                                                 |
| $ibeg$  | first index to consider in $P$                                                                  |
| $iend$  | last index to consider in $P$                                                                   |
| $A$     | input matrix                                                                                    |
| $lda$   | leading dimension of $A$                                                                        |
| $P$     | permutation in LAPACK format                                                                    |
| $psh$   | (optional): a sequential or parallel helper, to choose between sequential or parallel execution |

**Warning**

not sure the submatrix is still a permutation and the one we expect in all cases... examples for iend=2, ibeg=1 and P=[2,2,2]

**11.21.3.4 applyP() [2/4]**

```
template<class Field >
void applyP (
 const Field & F,
 const FFLAS::FFLAS_SIDE Side,
 const FFLAS::FFLAS_TRANSPOSE Trans,
 const size_t m,
 const size_t ibeg,
 const size_t iend,
 typename Field::Element_ptr A,
 const size_t lda,
 const size_t * P,
 const FFLAS::ParSeqHelper::Sequential seq) [inline]
```

**11.21.3.5 applyP() [3/4]**

```
template<class Field , class Cut , class Param >
void applyP (
 const Field & F,
 const FFLAS::FFLAS_SIDE Side,
 const FFLAS::FFLAS_TRANSPOSE Trans,
 const size_t m,
 const size_t ibeg,
 const size_t iend,
 typename Field::Element_ptr A,
 const size_t lda,
 const size_t * P,
 const FFLAS::ParSeqHelper::Parallel< Cut, Param > par) [inline]
```

**11.21.3.6 MonotonicApplyP()**

```
template<class Field >
void MonotonicApplyP (
 const Field & F,
 const FFLAS::FFLAS_SIDE Side,
 const FFLAS::FFLAS_TRANSPOSE Trans,
 const size_t M,
 const size_t ibeg,
 const size_t iend,
 typename Field::Element_ptr A,
 const size_t lda,
 const size_t * P,
 const size_t R) [inline]
```

Apply a R-monotonically increasing permutation P, to the matrix A.

MonotonicApplyP Apply a permutation defined by the first R entries of the vector P (the pivots).

The permutation represented by P is defined as follows:

- the first  $R$  values of  $P$  is a LAPACK representation (a sequence of transpositions)
- the remaining  $iend-ibeg-R$  values of the permutation are in a monotonically increasing progression  $Side==FFLAS::FflasLeft$  for row permutation  $Side==FFLAS::FflasRight$  for a column permutation  $Trans==FFLAS::FflasTrans$  for the inverse permutation of  $P$

## Parameters

|              |                                                                       |
|--------------|-----------------------------------------------------------------------|
| <i>F</i>     | base field                                                            |
| <i>Side</i>  | selects if it is a row (FflasLeft) or column (FflasRight) permutation |
| <i>Trans</i> | inverse permutation (FflasTrans/NoTrans)                              |
| <i>M</i>     |                                                                       |
| <i>ibeg</i>  |                                                                       |
| <i>iend</i>  |                                                                       |
| <i>A</i>     | input matrix                                                          |
| <i>lda</i>   | leading dimension of <i>A</i>                                         |
| <i>P</i>     | LAPACK permuation                                                     |
| <i>R</i>     | first values of $P$                                                   |

The non pivot elements, are located in montonically increasing order.

## 11.21.3.7 fgetrs() [1/4]

```
template<class Field >
void fgetrs (
 const Field & F,
 const FFLAS::FFLAS_SIDE Side,
 const size_t M,
 const size_t N,
 const size_t R,
 typename Field::Element_ptr A,
 const size_t lda,
 const size_t * P,
 const size_t * Q,
 typename Field::Element_ptr B,
 const size_t ldb,
 int * info)
```

Solve the system  $AX = B$  or  $XA = B$ .

Solving using the PLUQ decomposition of  $A$  already computed inplace with PLUQ (FFLAS::FflasNonUnit). Version for  $A$  square. If  $A$  is rank deficient, a solution is returned if the system is consistent, Otherwise an info is 1

## Parameters

|             |                                                                                    |
|-------------|------------------------------------------------------------------------------------|
| <i>F</i>    | base field                                                                         |
| <i>Side</i> | Determine wheter the resolution is left (FflasLeft) or right (FflasRight) looking. |
| <i>M</i>    | row dimension of $B$                                                               |
| <i>N</i>    | col dimension of $B$                                                               |
| <i>R</i>    | rank of $A$                                                                        |
| <i>A</i>    | input matrix                                                                       |
| <i>lda</i>  | leading dimension of $A$                                                           |
| <i>P</i>    | row permutation of the PLUQ decomposition of $A$                                   |

## Parameters

|             |                                                                                 |
|-------------|---------------------------------------------------------------------------------|
| <i>Q</i>    | column permutation of the PLUQ decomposition of A                               |
| <i>B</i>    | Right/Left hand side matrix. Initially stores B, finally stores the solution X. |
| <i>ldb</i>  | leading dimension of B                                                          |
| <i>info</i> | Success of the computation: 0 if successfull, >0 if system is inconsistent      |

## 11.21.3.8 fgetrs() [2/4]

```

template<class Field >
Field::Element_ptr fgetrs (
 const Field & F,
 const FFLAS::FFLAS_SIDE Side,
 const size_t M,
 const size_t N,
 const size_t NRHS,
 const size_t R,
 typename Field::Element_ptr A,
 const size_t lda,
 const size_t * P,
 const size_t * Q,
 typename Field::Element_ptr X,
 const size_t ldx,
 typename Field::ConstElement_ptr B,
 const size_t ldb,
 int * info)

```

Solve the system  $A X = B$  or  $X A = B$ .

Solving using the PLUQ decomposition of A already computed inplace with PLUQ(FFLAS::FflasNonUnit). Version for A rectangular. If A is rank deficient, a solution is returned if the system is consistent, Otherwise an info is 1

## Parameters

|             |                                                                                                             |
|-------------|-------------------------------------------------------------------------------------------------------------|
| <i>F</i>    | base field                                                                                                  |
| <i>Side</i> | Determine wheter the resolution is left (FflasLeft) or right (FflasRight) looking.                          |
| <i>M</i>    | row dimension of A                                                                                          |
| <i>N</i>    | col dimension of A                                                                                          |
| <i>NRHS</i> | number of columns (if Side = FFLAS::FflasLeft) or row (if Side = FFLAS::FflasRight) of the matrices X and B |
| <i>R</i>    | rank of A                                                                                                   |
| <i>A</i>    | input matrix                                                                                                |
| <i>lda</i>  | leading dimension of A                                                                                      |
| <i>P</i>    | row permutation of the PLUQ decomposition of A                                                              |
| <i>Q</i>    | column permutation of the PLUQ decomposition of A                                                           |
| <i>X</i>    | solution matrix                                                                                             |
| <i>ldx</i>  | leading dimension of X                                                                                      |
| <i>B</i>    | Right/Left hand side matrix.                                                                                |
| <i>ldb</i>  | leading dimension of B                                                                                      |
| <i>info</i> | Succes of the computation: 0 if successfull, >0 if system is inconsistent                                   |

**11.21.3.9 fgesv()** [1/4]

```
template<class Field >
size_t fgesv (
 const Field & F,
 const FFLAS::FFLAS_SIDE Side,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr B,
 const size_t ldb,
 int * info)
```

Square system solver.

**Parameters**

|             |                                                                                     |
|-------------|-------------------------------------------------------------------------------------|
| <i>F</i>    | The computation domain                                                              |
| <i>Side</i> | Determine wheter the resolution is left (FflasLeft) or right (FflasRight) looking   |
| <i>M</i>    | row dimension of B                                                                  |
| <i>N</i>    | col dimension of B                                                                  |
| <i>A</i>    | input matrix                                                                        |
| <i>lda</i>  | leading dimension of A                                                              |
| <i>B</i>    | Right/Left hand side matrix. Initially contains B, finally contains the solution X. |
| <i>ldb</i>  | leading dimension of B                                                              |
| <i>info</i> | Success of the computation: 0 if successfull, >0 if system is inconsistent          |

**Returns**

the rank of the system

Solve the system  $A X = B$  or  $X A = B$ . Version for A square. If A is rank deficient, a solution is returned if the system is consistent, Otherwise an info is 1

**11.21.3.10 fgesv()** [2/4]

```
template<class Field >
size_t fgesv (
 const Field & F,
 const FFLAS::FFLAS_SIDE Side,
 const size_t M,
 const size_t N,
 const size_t NRHS,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr X,
 const size_t ldX,
 typename Field::ConstElement_ptr B,
 const size_t ldb,
 int * info)
```

Rectangular system solver.

## Parameters

|             |                                                                                                             |
|-------------|-------------------------------------------------------------------------------------------------------------|
| <i>F</i>    | The computation domain                                                                                      |
| <i>Side</i> | Determine wheter the resolution is left (FflasLeft) or right (FflasRight) looking                           |
| <i>M</i>    | row dimension of A                                                                                          |
| <i>N</i>    | col dimension of A                                                                                          |
| <i>NRHS</i> | number of columns (if Side = FFLAS::FflasLeft) or row (if Side = FFLAS::FflasRight) of the matrices X and B |
| <i>A</i>    | input matrix                                                                                                |
| <i>lda</i>  | leading dimension of A                                                                                      |
| <i>B</i>    | Right/Left hand side matrix. Initially contains B, finally contains the solution X.                         |
| <i>ldb</i>  | leading dimension of B                                                                                      |
| <i>X</i>    |                                                                                                             |
| <i>ldx</i>  |                                                                                                             |
| <i>info</i> | Success of the computation: 0 if successfull, >0 if system is inconsistent                                  |

## Returns

the rank of the system

Solve the system  $A X = B$  or  $X A = B$ . Version for A square. If A is rank deficient, a solution is returned if the system is consistent, Otherwise an info is 1

11.21.3.11 **fttrtri()** [1/2]

```
template<class Field >
void fttrtri (
 const Field & F,
 const FFLAS::FFLAS_UPLO Uplo,
 const FFLAS::FFLAS_DIAG Diag,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 const size_t threshold = __FFLASFFPACK_FTRTRI_THRESHOLD)
```

Compute the inverse of a triangular matrix.

## Parameters

|             |                                                        |
|-------------|--------------------------------------------------------|
| <i>F</i>    | base field                                             |
| <i>Uplo</i> | whether the matrix is upper or lower triangular        |
| <i>Diag</i> | whether the matrix is unit diagonal (FflasUnit/NoUnit) |
| <i>N</i>    | input matrix order                                     |
| <i>A</i>    | the input matrix                                       |
| <i>lda</i>  | leading dimension of A                                 |

11.21.3.12 **trinv\_left()** [1/2]

```
template<class Field >
```



```

void trinv_left (
 const Field & F,
 const size_t N,
 typename Field::ConstElement_ptr L,
 const size_t ldl,
 typename Field::Element_ptr X,
 const size_t ldx)

```

### 11.21.3.13 ftrtrm() [1/2]

```

template<class Field >
void ftrtrm (
 const Field & F,
 const FFLAS::FFLAS_SIDE side,
 const FFLAS::FFLAS_DIAG diag,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda)

```

Compute the product of two triangular matrices of opposite shape.

Product UL or LU of the upper, resp lower triangular matrices U and L stored one above the other in the square matrix A.

#### Parameters

|             |                                                                      |
|-------------|----------------------------------------------------------------------|
| <i>F</i>    | base field                                                           |
| <i>Side</i> | set to FflasLeft to compute the product UL, FflasRight to compute LU |
| <i>diag</i> | whether the matrix U is unit diagonal (FflasUnit/NoUnit)             |
| <i>N</i>    | input matrix order                                                   |
| <i>A</i>    | the input matrix                                                     |
| <i>lda</i>  | leading dimension of A                                               |

### 11.21.3.14 ftrstr()

```

template<class Field >
void ftrstr (
 const Field & F,
 const FFLAS::FFLAS_SIDE side,
 const FFLAS::FFLAS_UPLO Uplo,
 const FFLAS::FFLAS_DIAG diagA,
 const FFLAS::FFLAS_DIAG diagB,
 const size_t N,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::Element_ptr B,
 const size_t ldb,
 const size_t threshold = __FFLASFFPACK_FTRSTR_THRESHOLD) [inline]

```

Solve a triangular system with a triangular right hand side of the same shape.

## Parameters

|              |                                                                                                                       |
|--------------|-----------------------------------------------------------------------------------------------------------------------|
| <i>F</i>     | base field                                                                                                            |
| <i>Side</i>  | set to FflasLeft to compute $U1^{-1} * U2$ or $L1^{-1} * L2$ , FflasRight to compute $U1 * U2^{-1}$ or $L1 * L2^{-1}$ |
| <i>Uplo</i>  | whether the matrix A is upper or lower triangular                                                                     |
| <i>diag1</i> | whether the matrix U1 or L2 is unit diagonal (FflasUnit/NoUnit)                                                       |
| <i>diag2</i> | whether the matrix U2 or L2 is unit diagonal (FflasUnit/NoUnit)                                                       |
| <i>N</i>     | order of the input matrices                                                                                           |
| <i>A</i>     | the input matrix to be inverted (U1 or L1)                                                                            |
| <i>lda</i>   | leading dimension of A                                                                                                |
| <i>B</i>     | the input right hand side (U2 or L2)                                                                                  |
| <i>ldb</i>   | leading dimension of B                                                                                                |

## 11.21.3.15 ftrssyr2k()

```
template<class Field >
void ftrssyr2k (
 const Field & F,
 const FFLAS::FFLAS_UPLO Uplo,
 const FFLAS::FFLAS_DIAG diagA,
 const size_t N,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::Element_ptr B,
 const size_t ldb,
 const size_t threshold = __FFLASFFPACK_FTRSSYR2K_THRESHOLD) [inline]
```

Solve a triangular system in a symmetric sum: find B upper/lower triangular such that  $A^T B + B^T A = C$  where C is symmetric.

C is overwritten by B.

## Parameters

|                |              |                                                                                                                       |
|----------------|--------------|-----------------------------------------------------------------------------------------------------------------------|
|                | <i>F</i>     | base field                                                                                                            |
|                | <i>Side</i>  | set to FflasLeft to compute $U1^{-1} * U2$ or $L1^{-1} * L2$ , FflasRight to compute $U1 * U2^{-1}$ or $L1 * L2^{-1}$ |
|                | <i>Uplo</i>  | whether the matrix A is upper or lower triangular                                                                     |
|                | <i>diagA</i> | whether the matrix A is unit diagonal (FflasUnit/NoUnit)                                                              |
|                | <i>N</i>     | order of the input matrices                                                                                           |
|                | <i>A</i>     | the input matrix                                                                                                      |
|                | <i>lda</i>   | leading dimension of A                                                                                                |
| <i>in, out</i> | <i>B</i>     | the input right hand side where the output is written                                                                 |
|                | <i>ldb</i>   | leading dimension of B                                                                                                |

## 11.21.3.16 fsytrf() [1/3]

```
template<class Field >
bool fsytrf (
```

```

const Field & F,
const FFLAS::FFLAS_UPLO UpLo,
const size_t N,
typename Field::Element_ptr A,
const size_t lda,
const size_t threshold = __FFLASFFPACK_FSYTRF_THRESHOLD)

```

Triangular factorization of symmetric matrices.

#### Parameters

|         |        |                                                                                          |
|---------|--------|------------------------------------------------------------------------------------------|
|         | $F$    | The computation domain                                                                   |
|         | $UpLo$ | Determine wheter to store the upper (FflasUpper) or lower (FflasLower) triangular factor |
|         | $N$    | order of the matrix A                                                                    |
| in, out | $A$    | input matrix                                                                             |
|         | $lda$  | leading dimension of A                                                                   |

#### Returns

false if the A does not have generic rank profile, making the computation fail.

Compute the a triangular factorization of the matrix A:  $A = L \times D \times L^T$  if UpLo = FflasLower or  $A = U^T \times D \times U$  otherwise. D is a diagonal matrix. The matrices L and U are unit diagonal lower (resp. upper) triangular and overwrite the input matrix A. The matrix D is stored on the diagonal of A, as the diagonal of L or U is known to be all ones. If A does not have generic rank profile, the LDLT or UTDU factorizations is not defined, and the algorithm returns false.

#### 11.21.3.17 fsytrf() [2/3]

```

template<class Field >
bool fsytrf (
 const Field & F,
 const FFLAS::FFLAS_UPLO UpLo,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 const FFLAS::ParSeqHelper::Sequential seq,
 const size_t threshold = __FFLASFFPACK_FSYTRF_THRESHOLD) [inline]

```

#### 11.21.3.18 fsytrf() [3/3]

```

template<class Field , class Cut , class Param >
bool fsytrf (
 const Field & F,
 const FFLAS::FFLAS_UPLO UpLo,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 const FFLAS::ParSeqHelper::Parallel< Cut, Param > par,
 const size_t threshold = __FFLASFFPACK_FSYTRF_THRESHOLD) [inline]

```

**11.21.3.19 fsytrf\_nonunit()** [1/3]

```
template<class Field >
bool fsytrf_nonunit (
 const Field & F,
 const FFLAS::FFLAS_UPLO UpLo,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr D,
 const size_t incD,
 const size_t threshold = __FFLASFFPACK_FSYTRF_THRESHOLD)
```

Triangular factorization of symmetric matrices.

**Parameters**

|         |        |                                                                                          |
|---------|--------|------------------------------------------------------------------------------------------|
|         | $F$    | The computation domain                                                                   |
|         | $UpLo$ | Determine wheter to store the upper (FflasUpper) or lower (FflasLower) triangular factor |
|         | $N$    | order of the matrix A                                                                    |
| in, out | $A$    | input matrix                                                                             |
| in, out | $D$    |                                                                                          |
|         | $lda$  | leading dimension of A                                                                   |

**Returns**

false if the A does not have generic rank profile, making the computation fail.

Compute the a triangular factorization of the matrix A:  $A = L \times D_{inv} \times L^T$  if UpLo = FflasLower or  $A = U^T \times D \times U$  otherwise. D is a diagonal matrix. The matrices L and U are lower (resp. upper) triangular and overwrite the input matrix A. The matrix D need to be stored separately, as the diagonal of L or U are not unit. If A does not have generic rank profile, the LDLT or UTDU factorizations is not defined, and the algorithm returns false.

**11.21.3.20 PLUQ()** [1/6]

```
template<class Field >
size_t PLUQ (
 const Field & F,
 const FFLAS::FFLAS_DIAG Diag,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t * P,
 size_t * Q) [inline]
```

Compute a PLUQ factorization of the given matrix.

Return its rank. The permutations P and Q are represented using LAPACK's convention.

**Parameters**

|     |            |
|-----|------------|
| $F$ | base field |
|-----|------------|

## Parameters

|             |                                                                        |
|-------------|------------------------------------------------------------------------|
| <i>Diag</i> | whether U should have a unit diagonal (FflasUnit) or not (FflasNoUnit) |
| <i>M</i>    | matrix row dimension                                                   |
| <i>N</i>    | matrix column dimension                                                |
| <i>A</i>    | input matrix                                                           |
| <i>lda</i>  | leading dimension of A                                                 |
| <i>P</i>    | the row permutation                                                    |
| <i>Q</i>    | the column permutation                                                 |

## Returns

the rank of A

## Bibliography

- Dumas J-G., Pernet C., and Sultan Z. *Simultaneous computation of the row and column rank profiles*, ISSAC'13, 2013

## 11.21.3.21 pPLUQ()

```
template<class Field >
size_t pPLUQ (
 const Field & F,
 const FFLAS::FFLAS_DIAG Diag,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t * P,
 size_t * Q) [inline]
```

## 11.21.3.22 PLUQ() [2/6]

```
template<class Field >
size_t PLUQ (
 const Field & F,
 const FFLAS::FFLAS_DIAG Diag,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t * P,
 size_t * Q,
 const FFLAS::ParSeqHelper::Sequential & PSHelper,
 size_t BCThreshold = __FFLASFFPACK_PLUQ_THRESHOLD) [inline]
```

**11.21.3.23 PLUQ()** [3/6]

```
template<class Field , class Cut , class Param >
size_t PLUQ (
 const Field & F,
 const FFLAS::FFLAS_DIAG Diag,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t * P,
 size_t * Q,
 const FFLAS::ParSeqHelper::Parallel< Cut, Param > & PSHelper)
```

**11.21.3.24 LUdivine()** [1/4]

```
template<class Field >
size_t LUdivine (
 const Field & F,
 const FFLAS::FFLAS_DIAG Diag,
 const FFLAS::FFLAS_TRANSPOSE trans,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const FFPACK_LU_TAG LuTag = FfpackSlabRecursive,
 const size_t cutoff = __FFLASFFPACK_LUDIVINE_THRESHOLD)
```

Compute the CUP or PLE factorization of the given matrix.

Using a block algorithm and return its rank. The permutations P and Q are represented using LAPACK's convention.

**Parameters**

|               |                                                                                                                             |
|---------------|-----------------------------------------------------------------------------------------------------------------------------|
| <i>F</i>      | base field                                                                                                                  |
| <i>Diag</i>   | whether the transformation matrix (U of the CUP, L of the PLE) should have a unit diagonal (FflasUnit) or not (FflasNoUnit) |
| <i>trans</i>  | whether to compute the CUP decomposition (FflasNoTrans) or the PLE decomposition (FflasTrans)                               |
| <i>M</i>      | matrix row dimension                                                                                                        |
| <i>N</i>      | matrix column dimension                                                                                                     |
| <i>A</i>      | input matrix                                                                                                                |
| <i>lda</i>    | leading dimension of A                                                                                                      |
| <i>P</i>      | the factor of CUP or PLE                                                                                                    |
| <i>Q</i>      | a permutation indicating the pivot position in the echelon form C or E in its first r positions                             |
| <i>LuTag</i>  | flag for setting the earling termination if the matrix is singular                                                          |
| <i>cutoff</i> | threshold to basecase                                                                                                       |

**Returns**

the rank of A

**Bibliography**

- Jeannerod C-P, Pernet, C. and Storjohann, A. *Rank-profile revealing Gaussian elimination and the CUP matrix decomposition*, J. of Symbolic Comp., 2013
- Pernet C, Brassel M *LUdivine, une divine factorisation LU*, 2002

**11.21.3.25 ColumnEchelonForm()** [1/3]

```
template<class Field >
size_t ColumnEchelonForm (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 bool transform = false,
 const FFPACK_LU_TAG LuTag = FfpackSlabRecursive) [inline]
```

Compute the Column Echelon form of the input matrix in-place.

If LuTag == FfpackTileRecursive, then after the computation  $A = [M \setminus V]$  such that  $AU = C$  is a column echelon decomposition of A, with  $U = P^T [V]$  and  $C = M + Q [I_r] [0 \text{ } I_{n-r}] [0]$  If LuTag == FfpackTileRecursive then  $A = [N \setminus V]$  such that the same holds with  $M = Q N$

$Q^T = Q^T$  If transform=false, the matrix V is not computed. See also test-colechelon for an example of use

**Parameters**

|    |             |                                                                                       |
|----|-------------|---------------------------------------------------------------------------------------|
|    | $F$         | base field                                                                            |
|    | $M$         | number of rows                                                                        |
|    | $N$         | number of columns                                                                     |
| in | $A$         | input matrix                                                                          |
|    | $lda$       | leading dimension of A                                                                |
|    | $P$         | the column permutation                                                                |
|    | $Qt$        | the row position of the pivots in the echelon form                                    |
|    | $transform$ | decides whether V is computed                                                         |
|    | $LuTag$     | chooses the elimination algorithm. SlabRecursive for LUdivine, TileRecursive for PLUQ |

**11.21.3.26 pColumnEchelonForm()**

```
template<class Field >
size_t pColumnEchelonForm (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 bool transform = false,
 size_t numthreads = 0,
 const FFPACK_LU_TAG LuTag = FfpackTileRecursive) [inline]
```

**11.21.3.27 ColumnEchelonForm()** [2/3]

```
template<class Field , class PSHelper >
size_t ColumnEchelonForm (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const FFPACK_LU_TAG LuTag,
 const PSHelper & psH) [inline]
```

**11.21.3.28 RowEchelonForm()** [1/3]

```
template<class Field >
size_t RowEchelonForm (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform = false,
 const FFPACK_LU_TAG LuTag = FfpackSlabRecursive) [inline]
```

Compute the Row Echelon form of the input matrix in-place.

If  $\text{LuTag} == \text{FfpackTileRecursive}$ , then after the computation  $A = [L \setminus M]$  such that  $XA = R$  is a row echelon decomposition of  $A$ , with  $X = [L \ 0] P$  and  $R = M + [I_r \ 0] Q^T [In-r]$  If  $\text{LuTag} == \text{FfpackTileRecursive}$  then  $A = [L \setminus N]$  such that the same holds with  $M = N Q^T Qt = Q^T$  If  $\text{transform} = \text{false}$ , the matrix  $L$  is not computed. See also `test-rowechelon` for an example of use

**Parameters**

|    |             |                                                                                       |
|----|-------------|---------------------------------------------------------------------------------------|
|    | $F$         | base field                                                                            |
|    | $M$         | number of rows                                                                        |
|    | $N$         | number of columns                                                                     |
| in | $A$         | the input matrix                                                                      |
|    | $lda$       | leading dimension of $A$                                                              |
|    | $P$         | the row permutation                                                                   |
|    | $Qt$        | the column position of the pivots in the echelon form                                 |
|    | $transform$ | decides whether $L$ is computed                                                       |
|    | $LuTag$     | chooses the elimination algorithm. SlabRecursive for LUdivine, TileRecursive for PLUQ |

**11.21.3.29 pRowEchelonForm()**

```
template<class Field >
size_t pRowEchelonForm (
```



```

const Field & F,
const size_t M,
const size_t N,
typename Field::Element_ptr A,
const size_t lda,
size_t * P,
size_t * Qt,
const bool transform = false,
size_t numthreads = 0,
const FFPACK_LU_TAG LuTag = FfpackTileRecursive) [inline]

```

### 11.21.3.30 RowEchelonForm() [2/3]

```

template<class Field , class PSHelper >
size_t RowEchelonForm (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const FFPACK_LU_TAG LuTag,
 const PSHelper & psH) [inline]

```

### 11.21.3.31 ReducedColumnEchelonForm() [1/3]

```

template<class Field >
size_t ReducedColumnEchelonForm (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform = false,
 const FFPACK_LU_TAG LuTag = FfpackSlabRecursive) [inline]

```

Compute the Reduced Column Echelon form of the input matrix in-place.

After the computation  $A = [V]$  such that  $AX = R$  is a reduced col echelon  $[M\ 0]$  decomposition of  $A$ , where  $X = P^T [V]$  and  $R = Q [I_r] [0\ I_{n-r}] [M\ 0]$   $Qt = Q^T$  If transform=false, the matrix  $X$  is not computed and the matrix  $A = R$

#### Parameters

|                                                             |             |                                                                                       |
|-------------------------------------------------------------|-------------|---------------------------------------------------------------------------------------|
|                                                             | $F$         | base field                                                                            |
|                                                             | $M$         | number of rows                                                                        |
|                                                             | $N$         | number of columns                                                                     |
| in                                                          | $A$         | input matrix                                                                          |
|                                                             | $lda$       | leading dimension of $A$                                                              |
|                                                             | $P$         | the column permutation                                                                |
|                                                             | $Qt$        | the row position of the pivots in the echelon form                                    |
|                                                             | $transform$ | decides whether $X$ is computed                                                       |
| Generated on Mon Nov 13 2023 00:00:00 for FFPACK by Doxygen |             |                                                                                       |
|                                                             | $LuTag$     | chooses the elimination algorithm. SlabRecursive for LUdivine, TileRecursive for PLUQ |

### 11.21.3.32 pReducedColumnEchelonForm()

```
template<class Field >
size_t pReducedColumnEchelonForm (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform = false,
 size_t numthreads = 0,
 const FFPACK_LU_TAG LuTag = FfpackTileRecursive) [inline]
```

### 11.21.3.33 ReducedColumnEchelonForm() [2/3]

```
template<class Field , class PSHelper >
size_t ReducedColumnEchelonForm (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const FFPACK_LU_TAG LuTag,
 const PSHelper & pSH) [inline]
```

### 11.21.3.34 ReducedRowEchelonForm() [1/3]

```
template<class Field >
size_t ReducedRowEchelonForm (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform = false,
 const FFPACK_LU_TAG LuTag = FfpackSlabRecursive) [inline]
```

Compute the Reduced Row Echelon form of the input matrix in-place.

After the computation  $A = [V1 \ M]$  such that  $X \ A = R$  is a reduced row echelon  $[V2 \ 0]$  decomposition of  $A$ , where  $X = [V1 \ 0] \ P$  and  $R = [I_r \ M] \ Q^T [V2 \ In-r] [0] \ Qt = Q^T$  If transform=false, the matrix  $X$  is not computed and the matrix  $A = R$

#### Parameters

|     |                |
|-----|----------------|
| $F$ | base field     |
| $M$ | number of rows |

## Parameters

|    |             |                                                                                       |
|----|-------------|---------------------------------------------------------------------------------------|
|    | $N$         | number of columns                                                                     |
| in | $A$         | input matrix                                                                          |
|    | $lda$       | leading dimension of $A$                                                              |
|    | $P$         | the row permutation                                                                   |
|    | $Qt$        | the column position of the pivots in the echelon form                                 |
|    | $transform$ | decides whether $X$ is computed                                                       |
|    | $LuTag$     | chooses the elimination algorithm. SlabRecursive for LUdivine, TileRecursive for PLUQ |

## 11.21.3.35 pReducedRowEchelonForm()

```
template<class Field >
size_t pReducedRowEchelonForm (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform = false,
 size_t numthreads = 0,
 const FFPACK_LU_TAG LuTag = FfpackTileRecursive) [inline]
```

## 11.21.3.36 ReducedRowEchelonForm() [2/3]

```
template<class Field , class PSHelper >
size_t ReducedRowEchelonForm (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const FFPACK_LU_TAG LuTag,
 const PSHelper & psH) [inline]
```

## 11.21.3.37 Invert() [1/4]

```
template<class Field >
Field::Element_ptr Invert (
 const Field & F,
 const size_t M,
 typename Field::Element_ptr A,
 const size_t lda,
 int & nullity)
```

Invert the given matrix in place or computes its nullity if it is singular.

An inplace  $2n^3$  algorithm is used.

## Parameters

|         |           |                               |
|---------|-----------|-------------------------------|
|         | $F$       | The computation domain        |
|         | $M$       | order of the matrix           |
| in, out | $A$       | input matrix ( $M \times M$ ) |
|         | $lda$     | leading dimension of A        |
|         | $nullity$ | dimension of the kernel of A  |

## Returns

pointer to  $A$  and  $A \leftarrow A^{-1}$

**11.21.3.38 Invert()** [2/4]

```
template<class Field >
Field::Element_ptr Invert (
 const Field & F,
 const size_t M,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::Element_ptr X,
 const size_t ldx,
 int & nullity)
```

Invert the given matrix or computes its nullity if it is singular.

## Precondition

$X$  is preallocated and should be large enough to store the  $m \times m$  matrix  $A$ .

## Parameters

|     |           |                                                                                                                  |
|-----|-----------|------------------------------------------------------------------------------------------------------------------|
|     | $F$       | The computation domain                                                                                           |
|     | $M$       | order of the matrix                                                                                              |
| in  | $A$       | input matrix ( $M \times M$ )                                                                                    |
|     | $lda$     | leading dimension of A                                                                                           |
| out | $X$       | this is the inverse of A if A is invertible (non NULL and <code>nullity = 0</code> ). It is untouched otherwise. |
|     | $ldx$     | leading dimension of X                                                                                           |
|     | $nullity$ | dimension of the kernel of A                                                                                     |

## Returns

pointer to  $X = A^{-1}$

**11.21.3.39 Invert2()** [1/2]

```
template<class Field >
Field::Element_ptr Invert2 (
```

```

const Field & F,
const size_t M,
typename Field::Element_ptr A,
const size_t lda,
typename Field::Element_ptr X,
const size_t ldx,
int & nullity)

```

Invert the given matrix or computes its nullity if it is singular.

An  $2n^3f$  algorithm is used. This routine can be % faster than [FFPACK::Invert](#) but is not totally inplace.

#### Precondition

X is preallocated and should be large enough to store the  $m \times m$  matrix A.

#### Warning

A is overwritten here !

**Bug** not tested.

#### Parameters

|         |           |                                                                                                                  |
|---------|-----------|------------------------------------------------------------------------------------------------------------------|
|         | $F$       | the computation domain                                                                                           |
|         | $M$       | order of the matrix                                                                                              |
| in, out | $A$       | input matrix ( $M \times M$ ). On output, A is modified and represents a "psycological" factorisation LU.        |
|         | $lda$     | leading dimension of A                                                                                           |
| out     | $X$       | this is the inverse of A if A is invertible (non NULL and <code>nullity = 0</code> ). It is untouched otherwise. |
|         | $ldx$     | leading dimension of X                                                                                           |
|         | $nullity$ | dimension of the kernel of A                                                                                     |

#### Returns

pointer to  $X = A^{-1}$

**Todo** this init is not all necessary (done after `fttrtri`)

**Todo** this init is not all necessary (done after `fttrtri`)

#### 11.21.3.40 CharPoly() [1/8]

```

template<class PolRing >
std::list< typename PolRing::Element > & CharPoly (
 const PolRing & R,
 std::list< typename PolRing::Element > & charp,
 const size_t N,

```

```

typename PolRing::Domain_t::Element_ptr A,
const size_t lda,
typename PolRing::Domain_t::RandIter & G,
const FFPACK_CHARPOLY_TAG CharpTag = FfpackAuto,
const size_t degree = __FFLASFFPACK_ARITHPROG_THRESHOLD) [inline]

```

Compute the characteristic polynomial of the matrix A.

#### Parameters

|     |                 |                                                                                       |
|-----|-----------------|---------------------------------------------------------------------------------------|
|     | <i>R</i>        | the polynomial ring of charp (contains the base field)                                |
| out | <i>charp</i>    | the characteristic polynomial of as a list of factors                                 |
|     | <i>N</i>        | order of the matrix A                                                                 |
| in  | <i>A</i>        | the input matrix ( $N \times N$ ) (could be overwritten in some algorithmic variants) |
|     | <i>lda</i>      | leading dimension of A                                                                |
|     | <i>CharpTag</i> | the algorithmic variant                                                               |
|     | <i>G</i>        | a random iterator (required for the randomized variants LUKrylov and ArithProg)       |

#### 11.21.3.41 CharPoly() [2/8]

```

template<class PolRing >
PolRing::Element & CharPoly (
 const PolRing & R,
 typename PolRing::Element & charp,
 const size_t N,
 typename PolRing::Domain_t::Element_ptr A,
 const size_t lda,
 typename PolRing::Domain_t::RandIter & G,
 const FFPACK_CHARPOLY_TAG CharpTag = FfpackAuto,
 const size_t degree = __FFLASFFPACK_ARITHPROG_THRESHOLD) [inline]

```

Compute the characteristic polynomial of the matrix A.

#### Parameters

|     |                 |                                                                                       |
|-----|-----------------|---------------------------------------------------------------------------------------|
|     | <i>R</i>        | the polynomial ring of charp (contains the base field)                                |
| out | <i>charp</i>    | the characteristic polynomial of as a single polynomial                               |
|     | <i>N</i>        | order of the matrix A                                                                 |
| in  | <i>A</i>        | the input matrix ( $N \times N$ ) (could be overwritten in some algorithmic variants) |
|     | <i>lda</i>      | leading dimension of A                                                                |
|     | <i>CharpTag</i> | the algorithmic variant                                                               |
|     | <i>G</i>        | a random iterator (required for the randomized variants LUKrylov and ArithProg)       |

#### 11.21.3.42 CharPoly() [3/8]

```

template<class PolRing >
PolRing::Element & CharPoly (
 const PolRing & R,
 typename PolRing::Element & charp,

```

```

const size_t N,
typename PolRing::Domain_t::Element_ptr A,
const size_t lda,
const FFPACK_CHARPOLY_TAG CharpTag = FfpackAuto,
const size_t degree = __FFLASFFPACK_ARITHPROG_THRESHOLD) [inline]

```

Compute the characteristic polynomial of the matrix A.

#### Parameters

|     |                 |                                                                                       |
|-----|-----------------|---------------------------------------------------------------------------------------|
|     | <i>R</i>        | the polynomial ring of charp (contains the base field)                                |
| out | <i>charp</i>    | the characteristic polynomial of A as a single polynomial                             |
|     | <i>N</i>        | order of the matrix A                                                                 |
| in  | <i>A</i>        | the input matrix ( $N \times N$ ) (could be overwritten in some algorithmic variants) |
|     | <i>lda</i>      | leading dimension of A                                                                |
|     | <i>CharpTag</i> | the algorithmic variant                                                               |

#### 11.21.3.43 MinPoly() [1/4]

```

template<class Field , class Polynomial >
Polynomial & MinPoly (
 const Field & F,
 Polynomial & minP,
 const size_t N,
 typename Field::ConstElement_ptr A,
 const size_t lda) [inline]

```

Compute the minimal polynomial of the matrix A.

The algorithm is randomized probabilistic, and computes the minimal polynomial of the Krylov iterates of a random vector: (v, Av, ..., A<sup>k</sup>v)

#### Parameters

|     |             |                                   |
|-----|-------------|-----------------------------------|
|     | <i>F</i>    | the base field                    |
| out | <i>minP</i> | the minimal polynomial of A       |
|     | <i>N</i>    | order of the matrix A             |
| in  | <i>A</i>    | the input matrix ( $N \times N$ ) |
|     | <i>lda</i>  | leading dimension of A            |

#### 11.21.3.44 MinPoly() [2/4]

```

template<class Field , class Polynomial , class RandIter >
Polynomial & MinPoly (
 const Field & F,
 Polynomial & minP,
 const size_t N,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 RandIter & G) [inline]

```

Compute the minimal polynomial of the matrix  $A$ .

The algorithm is randomized probabilistic, and computes the minimal polynomial of the Krylov iterates of a random vector:  $(v, Av, \dots, A^kv)$

#### Parameters

|     |        |                                   |
|-----|--------|-----------------------------------|
|     | $F$    | the base field                    |
| out | $minP$ | the minimal polynomial of $A$     |
|     | $N$    | order of the matrix $A$           |
| in  | $A$    | the input matrix ( $N \times N$ ) |
|     | $lda$  | leading dimension of $A$          |
|     | $G$    | a random iterator                 |

#### 11.21.3.45 MatVecMinPoly() [1/2]

```
template<class Field , class Polynomial >
Polynomial & MatVecMinPoly (
 const Field & F,
 Polynomial & minP,
 const size_t N,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr v,
 const size_t incv) [inline]
```

Compute the minimal polynomial of the matrix  $A$  and a vector  $v$ , namely the first linear dependency relation in the Krylov basis  $(v, Av, \dots, A^N v)$ .

#### Parameters

|     |        |                                                                                   |
|-----|--------|-----------------------------------------------------------------------------------|
|     | $F$    | the base field                                                                    |
| out | $minP$ | the minimal polynomial of $A$ and $v$                                             |
|     | $N$    | order of the matrix $A$                                                           |
| in  | $A$    | the input matrix ( $N \times N$ )                                                 |
|     | $lda$  | leading dimension of $A$                                                          |
|     | $K$    | an $N \times (N + 1)$ matrix containing the vector $v$ on its first row           |
|     | $ldk$  | leading dimension of $K$                                                          |
|     | $P$    | [out] (optional) the permutation used in the elimination of the Krylov matrix $K$ |

#### 11.21.3.46 Rank() [1/3]

```
template<class Field >
size_t Rank (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda)
```



Computes the rank of the given matrix using a PLUQ factorization.

The input matrix is modified.

#### Parameters

|    |       |                                                                               |
|----|-------|-------------------------------------------------------------------------------|
|    | $F$   | base field                                                                    |
|    | $M$   | row dimension of the matrix                                                   |
|    | $N$   | column dimension of the matrix                                                |
| in | $A$   | input matrix                                                                  |
|    | $lda$ | leading dimension of $A$                                                      |
|    | $psH$ | (optional) a ParSeqHelper to choose between sequential and parallel execution |

#### 11.21.3.47 pRank()

```
template<class Field >
size_t pRank (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t numthreads = 0)
```

#### 11.21.3.48 Rank() [2/3]

```
template<class Field , class PSHelper >
size_t Rank (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 const PSHelper & psH)
```

#### 11.21.3.49 IsSingular() [1/2]

```
template<class Field >
bool IsSingular (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda)
```

Returns true if the given matrix is singular.

The method is a block elimination with early termination

using LQUP factorization with early termination. If  $M \neq N$ , then the matrix is virtually padded with zeros to make it square and its determinant is zero.

#### Warning

The input matrix is modified.

## Parameters

|         |       |                                 |
|---------|-------|---------------------------------|
|         | $F$   | base field                      |
|         | $M$   | row dimension of the matrix     |
|         | $N$   | column dimension of the matrix. |
| in, out | $A$   | input matrix                    |
|         | $lda$ | leading dimension of A          |

## 11.21.3.50 Det() [1/6]

```
template<class Field >
Field::Element & Det (
 const Field & F,
 typename Field::Element & det,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t * P = NULL,
 size_t * Q = NULL) [inline]
```

Returns the determinant of the given square matrix.

The method is a block elimination using PLUQ factorization. The input matrix A is overwritten.

## Warning

The input matrix is modified.

## Parameters

|         |       |                                                                                                                                                             |
|---------|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
|         | $F$   | base field                                                                                                                                                  |
| out     | $det$ | the determinant of A                                                                                                                                        |
|         | $N$   | the order of the square matrix A.                                                                                                                           |
| in, out | $A$   | input matrix                                                                                                                                                |
|         | $lda$ | leading dimension of A                                                                                                                                      |
|         | $psH$ | (optional) a ParSeqHelper to choose between sequential and parallel execution                                                                               |
|         | $P,Q$ | (optional) row and column permutations to be used by the PLUQ factorization. randomized checkers (see cherckes/checker_det.inl) need them for certification |

## 11.21.3.51 pDet()

```
template<class Field >
Field::Element & pDet (
 const Field & F,
 typename Field::Element & det,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t numthreads = 0,
```

```

size_t * P = NULL,
size_t * Q = NULL) [inline]

```

#### 11.21.3.52 Det() [2/6]

```

template<class Field , class PSHelper >
Field::Element & Det (
 const Field & F,
 typename Field::Element & det,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 const PSHelper & pSH,
 size_t * P = NULL,
 size_t * Q = NULL)

```

#### 11.21.3.53 Solve() [1/3]

```

template<class Field >
Field::Element_ptr Solve (
 const Field & F,
 const size_t M,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr x,
 const int incx,
 typename Field::ConstElement_ptr b,
 const int incb) [inline]

```

Solves a linear system  $AX = b$  using PLUQ factorization.

@oaram F base field @oaram M matrix order

##### Parameters

|     |             |                                     |
|-----|-------------|-------------------------------------|
| in  | <i>A</i>    | input matrix                        |
|     | <i>lda</i>  | leading dimension of A              |
| out | <i>x</i>    | output solution vector              |
|     | <i>incx</i> | increment of x                      |
|     | <i>b</i>    | input right hand side of the system |
|     | <i>incb</i> | increment of b                      |

#### 11.21.3.54 Solve() [2/3]

```

template<class Field , class PSHelper >
Field::Element_ptr Solve (
 const Field & F,
 const size_t M,
 typename Field::Element_ptr A,
 const size_t lda,

```

```

typename Field::Element_ptr x,
const int incx,
typename Field::ConstElement_ptr b,
const int incb,
PSHelper & psH)

```

### 11.21.3.55 pSolve()

```

template<class Field >
Field::Element_ptr pSolve (
 const Field & F,
 const size_t M,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr x,
 const int incx,
 typename Field::ConstElement_ptr b,
 const int incb,
 size_t numthreads = 0) [inline]

```

### 11.21.3.56 RandomNullSpaceVector() [1/3]

```

template<class Field >
*void RandomNullSpaceVector (
 const Field & F,
 const FFLAS::FFLAS_SIDE Side,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr X,
 const size_t incX)

```

Solve  $LX = B$  or  $XL = B$  in place.

L is  $M \times M$  if Side == [FFLAS::FflasLeft](#) and  $N \times N$  if Side == [FFLAS::FflasRight](#), B is  $M \times N$ . Only the R non trivial column of L are stored in the  $M \times R$  matrix L Requirement : so that L could be expanded in-place Computes a vector of the Left/Right nullspace of the matrix A.

#### Parameters

|         |             |                                                                                  |
|---------|-------------|----------------------------------------------------------------------------------|
|         | <i>F</i>    | The computation domain                                                           |
|         | <i>Side</i> | decides whether it computes the left (FflasLeft) or right (FflasRight) nullspace |
|         | <i>M</i>    | number of rows                                                                   |
|         | <i>N</i>    | number of columns                                                                |
| in, out | <i>A</i>    | input matrix of dimension $M \times N$ , A is modified to its LU version         |
|         | <i>lda</i>  | leading dimension of A                                                           |
| out     | <i>X</i>    | output vector                                                                    |
|         | <i>incX</i> | increment of X                                                                   |

**11.21.3.57 NullSpaceBasis()** [1/2]

```
template<class Field >
size_t NullSpaceBasis (
 const Field & F,
 const FFLAS::FFLAS_SIDE Side,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr & NS,
 size_t & ldn,
 size_t & NSdim)
```

Computes a basis of the Left/Right nullspace of the matrix A.

return the dimension of the nullspace.

**Parameters**

|         |              |                                                                                  |
|---------|--------------|----------------------------------------------------------------------------------|
|         | <i>F</i>     | The computation domain                                                           |
|         | <i>Side</i>  | decides whether it computes the left (FflasLeft) or right (FflasRight) nullspace |
|         | <i>M</i>     | number of rows                                                                   |
|         | <i>N</i>     | number of columns                                                                |
| in, out | <i>A</i>     | input matrix of dimension M x N, A is modified                                   |
|         | <i>lda</i>   | leading dimension of A                                                           |
| out     | <i>NS</i>    | output matrix of dimension N x NSdim (allocated here)                            |
| out     | <i>ldn</i>   | leading dimension of NS                                                          |
| out     | <i>NSdim</i> | the dimension of the Nullspace (N-rank(A))                                       |

**11.21.3.58 RowRankProfile()** [1/3]

```
template<class Field >
size_t RowRankProfile (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t *& rkprofile,
 const FFPACK_LU_TAG LuTag = FfpackSlabRecursive) [inline]
```

Computes the row rank profile of A.

**Parameters**

|     |                  |                                                                                       |
|-----|------------------|---------------------------------------------------------------------------------------|
|     | <i>F</i>         | base field                                                                            |
|     | <i>M</i>         | number of rows                                                                        |
|     | <i>N</i>         | number of columns                                                                     |
| in  | <i>A</i>         | input matrix of dimension M x N                                                       |
|     | <i>lda</i>       | leading dimension of A                                                                |
| out | <i>rkprofile</i> | return the rank profile as an array of row indexes, of dimension r=rank(A)            |
|     | <i>LuTag</i>     | chooses the elimination algorithm. SlabRecursive for LUdivine, TileRecursive for PLUQ |

A modified rkprofile is allocated during the computation.

Returns

R

### 11.21.3.59 pRowRankProfile()

```
template<class Field >
size_t pRowRankProfile (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t *rkprofile,
 size_t numthreads = 0,
 const FFPACK_LU_TAG LuTag = FfpackTileRecursive) [inline]
```

### 11.21.3.60 RowRankProfile() [2/3]

```
template<class Field , class PSHelper >
size_t RowRankProfile (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t *rkprofile,
 const FFPACK_LU_TAG LuTag,
 PSHelper & psH) [inline]
```

### 11.21.3.61 ColumnRankProfile() [1/3]

```
template<class Field >
size_t ColumnRankProfile (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t *rkprofile,
 const FFPACK_LU_TAG LuTag = FfpackSlabRecursive) [inline]
```

Computes the column rank profile of A.

Parameters

|     |             |                                                                                       |
|-----|-------------|---------------------------------------------------------------------------------------|
|     | $F$         | base field                                                                            |
|     | $M$         | number of rows                                                                        |
|     | $N$         | number of columns                                                                     |
| in  | $A$         | input matrix of dimension                                                             |
|     | $lda$       | leading dimension of A                                                                |
| out | $rkprofile$ | return the rank profile as an array of row indexes, of dimension $r=\text{rank}(A)$   |
|     | $LuTag$     | chooses the elimination algorithm. SlabRecursive for LUdivine, TileRecursive for PLUQ |

A modified rkprofile is allocated during the computation.

Returns

R

### 11.21.3.62 pColumnRankProfile()

```
template<class Field >
size_t pColumnRankProfile (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t *rkprofile,
 size_t numthreads = 0,
 const FFPACK_LU_TAG LuTag = FfpackTileRecursive) [inline]
```

### 11.21.3.63 ColumnRankProfile() [2/3]

```
template<class Field , class PSHelper >
size_t ColumnRankProfile (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t *rkprofile,
 const FFPACK_LU_TAG LuTag,
 PSHelper & psH) [inline]
```

### 11.21.3.64 RankProfileFromLU()

```
void RankProfileFromLU (
 const size_t * P,
 const size_t N,
 const size_t R,
 size_t * rkprofile,
 const FFPACK_LU_TAG LuTag) [inline]
```

Recovers the column/row rank profile from the permutation of an LU decomposition.

Works with both the CUP/PLE decompositions (obtained by LUdivine) or the PLUQ decomposition. Assumes that the output vector containing the rank profile is already allocated.

Parameters

|     |             |                                                                                       |
|-----|-------------|---------------------------------------------------------------------------------------|
|     | $P$         | the permutation carrying the rank profile information                                 |
|     | $N$         | the row/col dimension for a row/column rank profile                                   |
|     | $R$         | the rank of the matrix                                                                |
| out | $rkprofile$ | return the rank profile as an array of indices                                        |
|     | $LuTag$     | chooses the elimination algorithm. StabRecursive for LUdivine, TileRecursive for PLUQ |

### 11.21.3.65 LeadingSubmatrixRankProfiles()

```
size_t LeadingSubmatrixRankProfiles (
 const size_t M,
 const size_t N,
 const size_t R,
 const size_t LSm,
 const size_t LSn,
 const size_t * P,
 const size_t * Q,
 size_t * RRP,
 size_t * CRP) [inline]
```

Recovers the row and column rank profiles of any leading submatrix from the PLUQ decomposition.

Only works with the PLUQ decomposition Assumes that the output vectors containing the rank profiles are already allocated.

#### Parameters

|       |                                                          |
|-------|----------------------------------------------------------|
| $P$   | the permutation carrying the rank profile information    |
| $M$   | the row dimension of the initial matrix                  |
| $N$   | the column dimension of the initial matrix               |
| $R$   | the rank of the initial matrix                           |
| $LSm$ | the row dimension of the leading submatrix considered    |
| $LSn$ | the column dimension of the leading submatrix considered |
| $P$   | the row permutation of the PLUQ decomposition            |
| $Q$   | the column permutation of the PLUQ decomposition         |
| $RRP$ | return the row rank profile of the leading submatrix     |

#### Returns

the rank of the  $LSm \times LSn$  leading submatrix

A is modified

**Bibliography** • Dumas J-G., Pernet C., and Sultan Z. *Simultaneous computation of the row and column rank profiles*, ISSAC'13.

### 11.21.3.66 RowRankProfileSubmatrixIndices() [1/2]

```
template<class Field >
size_t RowRankProfileSubmatrixIndices (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t *& rowindices,
 size_t *& colindices,
 size_t & R)
```

RowRankProfileSubmatrixIndices.

Computes the indices of the submatrix  $r \times r$  X of A whose rows correspond to the row rank profile of A.



## Parameters

|     |                   |                                    |
|-----|-------------------|------------------------------------|
|     | $F$               | base field                         |
|     | $M$               | number of rows                     |
|     | $N$               | number of columns                  |
| in  | $A$               | input matrix of dimension          |
|     | <i>rowindices</i> | array of the row indices of X in A |
|     | <i>colindices</i> | array of the col indices of X in A |
|     | <i>lda</i>        | leading dimension of A             |
| out | $R$               | list of indices                    |

rowindices and colindices are allocated during the computation. A is modified

## Returns

$R$

## 11.21.3.67 ColRankProfileSubmatrixIndices() [1/2]

```
template<class Field >
size_t ColRankProfileSubmatrixIndices (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t *& rowindices,
 size_t *& colindices,
 size_t & R)
```

Computes the indices of the submatrix  $r \times r$  X of A whose columns correspond to the column rank profile of A.

## Parameters

|     |                   |                                    |
|-----|-------------------|------------------------------------|
|     | $F$               | base field                         |
|     | $M$               | number of rows                     |
|     | $N$               | number of columns                  |
| in  | $A$               | input matrix of dimension          |
|     | <i>rowindices</i> | array of the row indices of X in A |
|     | <i>colindices</i> | array of the col indices of X in A |
|     | <i>lda</i>        | leading dimension of A             |
| out | $R$               | list of indices                    |

rowindices and colindices are allocated during the computation.

## Warning

A is modified

## Returns

R

**11.21.3.68 RowRankProfileSubmatrix()** [1/2]

```
template<class Field >
size_t RowRankProfileSubmatrix (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr & X,
 size_t & R)
```

Computes the  $r \times r$  submatrix  $X$  of  $A$ , by picking the row rank profile rows of  $A$ .

## Parameters

|     |       |                                        |
|-----|-------|----------------------------------------|
|     | $F$   | base field                             |
|     | $M$   | number of rows                         |
|     | $N$   | number of columns                      |
| in  | $A$   | input matrix of dimension $M \times N$ |
|     | $lda$ | leading dimension of $A$               |
| out | $X$   | the output matrix                      |
| out | $R$   | list of indices                        |

$A$  is not modified  $X$  is allocated during the computation.

## Returns

R

**11.21.3.69 ColRankProfileSubmatrix()** [1/2]

```
template<class Field >
size_t ColRankProfileSubmatrix (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr & X,
 size_t & R)
```

Compute the  $r \times r$  submatrix  $X$  of  $A$ , by picking the row rank profile rows of  $A$ .

## Parameters

|  |     |                |
|--|-----|----------------|
|  | $F$ | base field     |
|  | $M$ | number of rows |

## Parameters

|     |       |                                        |
|-----|-------|----------------------------------------|
|     | $N$   | number of columns                      |
| in  | $A$   | input matrix of dimension $M \times N$ |
|     | $lda$ | leading dimension of $A$               |
| out | $X$   | the output matrix                      |
| out | $R$   | list of indices                        |

$A$  is not modified  $X$  is allocated during the computation.

## Returns

$R$

## 11.21.3.70 getTriangular() [1/2]

```
template<class Field >
void getTriangular (
 const Field & F,
 const FFLAS::FFLAS_UPLO Uplo,
 const FFLAS::FFLAS_DIAG diag,
 const size_t M,
 const size_t N,
 const size_t R,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::Element_ptr T,
 const size_t ldt,
 const bool OnlyNonZeroVectors = false) [inline]
```

Extracts a triangular matrix from a compact storage  $A=L\backslash U$  of rank  $R$ .

if OnlyNonZeroVectors is false, then  $T$  and  $A$  have the same dimensions Otherwise,  $T$  is  $R \times N$  if UpLo = FflasUpper, else  $T$  is  $M \times R$

## Parameters

|     |                      |                                                                                       |
|-----|----------------------|---------------------------------------------------------------------------------------|
|     | $F$                  | base field                                                                            |
|     | $UpLo$               | selects if the upper (FflasUpper) or lower (FflasLower) triangular matrix is returned |
|     | $diag$               | selects if the triangular matrix unit-diagonal (FflasUnit/NoUnit)                     |
|     | $M$                  | row dimension of $T$                                                                  |
|     | $N$                  | column dimension of $T$                                                               |
|     | $R$                  | rank of the triangular matrix (how many rows/columns need to be copied)               |
| in  | $A$                  | input matrix                                                                          |
|     | $lda$                | leading dimension of $A$                                                              |
| out | $T$                  | output matrix                                                                         |
|     | $ldt$                | leading dimension of $T$                                                              |
|     | $OnlyNonZeroVectors$ | decides whether the last zero rows/columns should be ignored                          |

**Todo** just one triangular fzero+fassign ?

**Todo** just one triangular fzero+fassign ?

### 11.21.3.71 getTriangular() [2/2]

```
template<class Field >
void getTriangular (
 const Field & F,
 const FFLAS::FFLAS_UPLO Uplo,
 const FFLAS::FFLAS_DIAG diag,
 const size_t M,
 const size_t N,
 const size_t R,
 typename Field::Element_ptr A,
 const size_t lda) [inline]
```

Cleans up a compact storage  $A=LU$  to reveal a triangular matrix of rank  $R$ .

#### Parameters

|                |             |                                                                                       |
|----------------|-------------|---------------------------------------------------------------------------------------|
|                | <i>F</i>    | base field                                                                            |
|                | <i>UpLo</i> | selects if the upper (FflasUpper) or lower (FflasLower) triangular matrix is revealed |
|                | <i>diag</i> | selects if the triangular matrix unit-diagonal (FflasUnit/NoUnit)                     |
|                | <i>M</i>    | row dimension of A                                                                    |
|                | <i>N</i>    | column dimension of A                                                                 |
|                | <i>R</i>    | rank of the triangular matrix                                                         |
| <i>in, out</i> | <i>A</i>    | input/output matrix                                                                   |
|                | <i>lda</i>  | leading dimension of A                                                                |

**Todo** just one triangular fzero+fassign ?

**Todo** just one triangular fzero+fassign ?

### 11.21.3.72 getEchelonForm() [1/2]

```
template<class Field >
void getEchelonForm (
 const Field & F,
 const FFLAS::FFLAS_UPLO Uplo,
 const FFLAS::FFLAS_DIAG diag,
 const size_t M,
 const size_t N,
 const size_t R,
 const size_t * P,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::Element_ptr T,
```

```

const size_t ldt,
const bool OnlyNonZeroVectors = false,
const FFPACK_LU_TAG LuTag = FfpackSlabRecursive) [inline]

```

Extracts a matrix in echelon form from a compact storage  $A=L\backslash U$  of rank  $R$  obtained by RowEchelonForm or ColumnEchelonForm.

Either  $L$  or  $U$  is in Echelon form (depending on Uplo) The echelon structure is defined by the first  $R$  values of the array  $P$ . row and column dimension of  $T$  are greater or equal to that of  $A$

#### Parameters

|     |                      |                                                                                       |
|-----|----------------------|---------------------------------------------------------------------------------------|
|     | $F$                  | base field                                                                            |
|     | $UpLo$               | selects if the upper (FflasUpper) or lower (FflasLower) triangular matrix is returned |
|     | $diag$               | selects if the echelon matrix has unit pivots (FflasUnit/NoUnit)                      |
|     | $M$                  | row dimension of $T$                                                                  |
|     | $N$                  | column dimension of $T$                                                               |
|     | $R$                  | rank of the triangular matrix (how many rows/columns need to be copied)               |
|     | $P$                  | positions of the $R$ pivots                                                           |
| in  | $A$                  | input matrix                                                                          |
|     | $lda$                | leading dimension of $A$                                                              |
| out | $T$                  | output matrix                                                                         |
|     | $ldt$                | leading dimension of $T$                                                              |
|     | $OnlyNonZeroVectors$ | decides whether the last zero rows/columns should be ignored                          |
|     | $LuTag$              | which factorized form (CUP/PLE if FfpackSlabRecursive, PLUQ if FfpackTileRecursive)   |

#### 11.21.3.73 getEchelonForm() [2/2]

```

template<class Field >
void getEchelonForm (
 const Field & F,
 const FFLAS::FFLAS_UPLO Uplo,
 const FFLAS::FFLAS_DIAG diag,
 const size_t M,
 const size_t N,
 const size_t R,
 const size_t * P,
 typename Field::Element_ptr A,
 const size_t lda,
 const FFPACK_LU_TAG LuTag = FfpackSlabRecursive) [inline]

```

Cleans up a compact storage  $A=L\backslash U$  obtained by RowEchelonForm or ColumnEchelonForm to reveal an echelon form of rank  $R$ .

Either  $L$  or  $U$  is in Echelon form (depending on Uplo) The echelon structure is defined by the first  $R$  values of the array  $P$ .

#### Parameters

|  |        |                                                                                       |
|--|--------|---------------------------------------------------------------------------------------|
|  | $F$    | base field                                                                            |
|  | $UpLo$ | selects if the upper (FflasUpper) or lower (FflasLower) triangular matrix is returned |

## Parameters

|                |              |                                                                                     |
|----------------|--------------|-------------------------------------------------------------------------------------|
|                | <i>diag</i>  | selects if the echelon matrix has unit pivots (FflasUnit/NoUnit)                    |
|                | <i>M</i>     | row dimension of A                                                                  |
|                | <i>N</i>     | column dimension of A                                                               |
|                | <i>R</i>     | rank of the triangular matrix (how many rows/columns need to be copied)             |
|                | <i>P</i>     | positions of the R pivots                                                           |
| <i>in, out</i> | <i>A</i>     | input/output matrix                                                                 |
|                | <i>lda</i>   | leading dimension of A                                                              |
|                | <i>LuTag</i> | which factorized form (CUP/PLE if FfpackSlabRecursive, PLUQ if FfpackTileRecursive) |

11.21.3.74 `getEchelonTransform()`

```
template<class Field >
void getEchelonTransform (
 const Field & F,
 const FFLAS::FFLAS_UPLO Uplo,
 const FFLAS::FFLAS_DIAG diag,
 const size_t M,
 const size_t N,
 const size_t R,
 const size_t * P,
 const size_t * Q,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::Element_ptr T,
 const size_t ldt,
 const FFPACK_LU_TAG LuTag = FfpackSlabRecursive) [inline]
```

Extracts a transformation matrix to echelon form from a compact storage  $A=LU$  of rank  $R$  obtained by Row↔EchelonForm or ColumnEchelonForm.

If Uplo == FflasLower:  $T$  is  $N \times N$  (already allocated) such that  $A T = C$  is a transformation of  $A$  in Column echelon form Else  $T$  is  $M \times M$  (already allocated) such that  $T A = E$  is a transformation of  $A$  in Row Echelon form

## Parameters

|            |              |                                                                                                         |
|------------|--------------|---------------------------------------------------------------------------------------------------------|
|            | <i>F</i>     | base field                                                                                              |
|            | <i>UpLo</i>  | Lower (FflasLower) means Transformation to Column Echelon Form, Upper (FflasUpper), to Row Echelon Form |
|            | <i>diag</i>  | selects if the echelon matrix has unit pivots (FflasUnit/NoUnit)                                        |
|            | <i>M</i>     | row dimension of A                                                                                      |
|            | <i>N</i>     | column dimension of A                                                                                   |
|            | <i>R</i>     | rank of the triangular matrix                                                                           |
|            | <i>P</i>     | permutation matrix                                                                                      |
| <i>in</i>  | <i>A</i>     | input matrix                                                                                            |
|            | <i>lda</i>   | leading dimension of A                                                                                  |
| <i>out</i> | <i>T</i>     | output matrix                                                                                           |
|            | <i>ldt</i>   | leading dimension of T                                                                                  |
|            | <i>LuTag</i> | which factorized form (CUP/PLE if FfpackSlabRecursive, PLUQ if FfpackTileRecursive)                     |

**11.21.3.75 getReducedEchelonForm()** [1/2]

```

template<class Field >
void getReducedEchelonForm (
 const Field & F,
 const FFLAS::FFLAS_UPLO Uplo,
 const size_t M,
 const size_t N,
 const size_t R,
 const size_t * P,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::Element_ptr T,
 const size_t ldt,
 const bool OnlyNonZeroVectors = false,
 const FFPACK_LU_TAG LuTag = FfpackSlabRecursive) [inline]

```

Extracts a matrix in echelon form from a compact storage  $A=L\backslash U$  of rank  $R$  obtained by ReducedRowEchelonForm or ReducedColumnEchelonForm with transform = true.

Either L or U is in Echelon form (depending on Uplo) The echelon structure is defined by the first  $R$  values of the array  $P$ . row and column dimension of  $T$  are greater or equal to that of  $A$

**Parameters**

|    |                      |                                                                                       |
|----|----------------------|---------------------------------------------------------------------------------------|
|    | $F$                  | base field                                                                            |
|    | $UpLo$               | selects if the upper (FflasUpper) or lower (FflasLower) triangular matrix is returned |
|    | $diag$               | selects if the echelon matrix has unit pivots (FflasUnit/NoUnit)                      |
|    | $M$                  | row dimension of $T$                                                                  |
|    | $N$                  | column dimension of $T$                                                               |
|    | $R$                  | rank of the triangular matrix (how many rows/columns need to be copied)               |
|    | $P$                  | positions of the $R$ pivots                                                           |
| in | $A$                  | input matrix                                                                          |
|    | $lda$                | leading dimension of $A$                                                              |
|    | $ldt$                | leading dimension of $T$                                                              |
|    | $LuTag$              | which factorized form (CUP/PLE if FfpackSlabRecursive, PLUQ if FfpackTileRecursive)   |
|    | $OnlyNonZeroVectors$ | decides whether the last zero rows/columns should be ignored                          |

**11.21.3.76 getReducedEchelonForm()** [2/2]

```

template<class Field >
void getReducedEchelonForm (
 const Field & F,
 const FFLAS::FFLAS_UPLO Uplo,
 const size_t M,
 const size_t N,
 const size_t R,
 const size_t * P,
 typename Field::Element_ptr A,
 const size_t lda,
 const FFPACK_LU_TAG LuTag = FfpackSlabRecursive) [inline]

```

Cleans up a compact storage  $A=L\backslash U$  of rank  $R$  obtained by `ReducedRowEchelonForm` or `ReducedColumnEchelonForm` with `transform = true`.

Either  $L$  or  $U$  is in Echelon form (depending on `Uplo`) The echelon structure is defined by the first  $R$  values of the array  $P$ .

#### Parameters

|           |         |                                                                                                                                            |
|-----------|---------|--------------------------------------------------------------------------------------------------------------------------------------------|
|           | $F$     | base field                                                                                                                                 |
|           | $UpLo$  | selects if the upper ( <code>FflasUpper</code> ) or lower ( <code>FflasLower</code> ) triangular matrix is returned                        |
|           | $diag$  | selects if the echelon matrix has unit pivots ( <code>FflasUnit/NoUnit</code> )                                                            |
|           | $M$     | row dimension of $A$                                                                                                                       |
|           | $N$     | column dimension of $A$                                                                                                                    |
|           | $R$     | rank of the triangular matrix (how many rows/columns need to be copied)                                                                    |
|           | $P$     | positions of the $R$ pivots                                                                                                                |
| $in, out$ | $A$     | input/output matrix                                                                                                                        |
|           | $lda$   | leading dimension of $A$                                                                                                                   |
|           | $LuTag$ | which factorized form ( <code>CUP/PLE</code> if <code>FfpackSlabRecursive</code> , <code>PLUQ</code> if <code>FfpackTileRecursive</code> ) |

### 11.21.3.77 `getReducedEchelonTransform()`

```
template<class Field >
void getReducedEchelonTransform (
 const Field & F,
 const FFLAS::FFLAS_UPLO Uplo,
 const size_t M,
 const size_t N,
 const size_t R,
 const size_t * P,
 const size_t * Q,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::Element_ptr T,
 const size_t ldt,
 const FFPACK_LU_TAG LuTag = FfpackSlabRecursive) [inline]
```

Extracts a transformation matrix to echelon form from a compact storage  $A=L\backslash U$  of rank  $R$  obtained by `RowEchelonForm` or `ColumnEchelonForm`.

If `Uplo == FflasLower`:  $T$  is  $N \times N$  (already allocated) such that  $A T = C$  is a transformation of  $A$  in Column echelon form Else  $T$  is  $M \times M$  (already allocated) such that  $T A = E$  is a transformation of  $A$  in Row Echelon form

#### Parameters

|      |        |                                                                                         |
|------|--------|-----------------------------------------------------------------------------------------|
|      | $F$    | base field                                                                              |
|      | $UpLo$ | selects Col ( <code>FflasLower</code> ) or Row ( <code>FflasUpper</code> ) Echelon Form |
|      | $diag$ | selects if the echelon matrix has unit pivots ( <code>FflasUnit/NoUnit</code> )         |
|      | $M$    | row dimension of $A$                                                                    |
|      | $N$    | column dimension of $A$                                                                 |
|      | $R$    | rank of the triangular matrix                                                           |
|      | $P$    | permutation matrix                                                                      |
| $in$ | $A$    | input matrix                                                                            |



## Parameters

|     |              |                                                                                     |
|-----|--------------|-------------------------------------------------------------------------------------|
|     | <i>lda</i>   | leading dimension of A                                                              |
| out | <i>T</i>     | output matrix                                                                       |
|     | <i>ldt</i>   | leading dimension of T                                                              |
|     | <i>LuTag</i> | which factorized form (CUP/PLE if FfpackSlabRecursive, PLUQ if FfpackTileRecursive) |

**11.21.3.78 PLUQtoEchelonPermutation()**

```
void PLUQtoEchelonPermutation (
 const size_t N,
 const size_t R,
 const size_t * P,
 size_t * outPerm) [inline]
```

Auxiliary routine: determines the permutation that changes a PLUQ decomposition into a echelon form revealing PLUQ decomposition.

**11.21.3.79 LTBruhatGen()**

```
template<class Field >
size_t LTBruhatGen (
 const Field & Fi,
 const FFLAS::FFLAS_DIAG diag,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t * P,
 size_t * Q) [inline]
```

LTBruhatGen Suppose A is Left Triangular Matrix This procedure computes the Bruhat Representation of A and return the rank of A.

## Parameters

|             |                                                |
|-------------|------------------------------------------------|
| <i>Fi</i>   | base Field                                     |
| <i>diag</i> |                                                |
| <i>N</i>    | size of A                                      |
| <i>A</i>    | the matrix we search the Bruhat representation |
| <i>lda</i>  | the leading dimension of A                     |
| <i>P</i>    | a permutation matrix                           |
| <i>Q</i>    | a permutation matrix                           |

**11.21.3.80 getLTBruhatGen() [1/2]**

```
template<class Field >
void getLTBruhatGen (
 const Field & Fi,
```

```

const size_t N,
const size_t r,
const size_t * P,
const size_t * Q,
typename Field::Element_ptr R,
const size_t ldr) [inline]

```

**GetLTBruhatGen** This procedure Computes the Rank Revealing Matrix based on the Bruhta representation of a Matrix.

#### Parameters

|            |                                                        |
|------------|--------------------------------------------------------|
| <i>Fi</i>  | base Field                                             |
| <i>N</i>   | size of the matrix                                     |
| <i>r</i>   | the rank of the matrix                                 |
| <i>P</i>   | a permutation matrix                                   |
| <i>Q</i>   | a permutation matrix                                   |
| <i>R</i>   | the matrix that will contain the rank revealing matrix |
| <i>ldr</i> | the leading fimension of R                             |

#### 11.21.3.81 getLTBruhatGen() [2/2]

```

template<class Field >
void getLTBruhatGen (
 const Field & Fi,
 const FFLAS::FFLAS_UPLO Uplo,
 const FFLAS::FFLAS_DIAG diag,
 const size_t N,
 const size_t r,
 const size_t * P,
 const size_t * Q,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::Element_ptr T,
 const size_t ldt) [inline]

```

**GetLTBruhatGen** This procedure computes the matrix L or U f the Bruhat Representation Suppose that A is the bruhat representation of a matrix.

#### Parameters

|             |                                       |
|-------------|---------------------------------------|
| <i>Fi</i>   | base Field                            |
| <i>Uplo</i> | choose if the procedure return L or U |
| <i>diag</i> |                                       |
| <i>N</i>    | size of A                             |
| <i>r</i>    | rank of A                             |
| <i>P</i>    | permutaion matrix                     |
| <i>Q</i>    | permutation matrix                    |
| <i>A</i>    | a bruhat representation               |
| <i>lda</i>  | leading dimension of A                |
| <i>T</i>    | matrix that will contains L or U      |
| <i>ldt</i>  | leading dimension of T                |

## 11.21.3.82 LTQSorter()

```
size_t LTQSorter (
 const size_t N,
 const size_t r,
 const size_t * P,
 const size_t * Q) [inline]
```

LTQSorter This procedure computes the order of quasiseparability of a matrix.

## Parameters

|     |                    |
|-----|--------------------|
| $N$ | size of the matrix |
| $r$ | rank of the matrix |
| $P$ | permutation matrix |
| $Q$ | permutation matrix |

## 11.21.3.83 CompressToBlockBiDiagonal()

```
template<class Field >
size_t CompressToBlockBiDiagonal (
 const Field & Fi,
 const FFLAS::FFLAS_UPLO Uplo,
 size_t N,
 size_t s,
 size_t r,
 const size_t * P,
 const size_t * Q,
 typename Field::Element_ptr A,
 size_t lda,
 typename Field::Element_ptr X,
 size_t ldx,
 size_t * K,
 size_t * M,
 size_t * T) [inline]
```

CompressToBlockBiDiagonal This procedure compress a compact representation of a row echelon form or column echelon form.

## Parameters

|        |                                                   |
|--------|---------------------------------------------------|
| $Fi$   | base Field                                        |
| $Uplo$ | chosse if the procedure is based on row or column |
| $N$    | size of the matrix                                |
| $s$    | order of quasiseparability                        |
| $r$    | rank                                              |
| $P$    | permutation matrix                                |
| $Q$    | permutation matrix                                |
| $A$    | the matrix to compact                             |
| $lda$  | leading dimension of A                            |
| $X$    | matrix that will stock the representation         |
| $ldx$  | leading dimension of X                            |

## Parameters

|     |                                           |
|-----|-------------------------------------------|
| $K$ | stock the position of the blocks in A     |
| $M$ | permutation matrix                        |
| $T$ | stock the operation done in the procedure |

**11.21.3.84 ExpandBlockBiDiagonalToBruhat()**

```
template<class Field >
void ExpandBlockBiDiagonalToBruhat (
 const Field & Fi,
 const FFLAS::FFLAS_UPLO Uplo,
 size_t N,
 size_t s,
 size_t r,
 typename Field::Element_ptr A,
 size_t lda,
 typename Field::Element_ptr X,
 size_t ldx,
 size_t NbBlocks,
 size_t * K,
 size_t * M,
 size_t * T) [inline]
```

ExpandBlockBiDiagonal This procedure expand a compact representation of a row echelon form or column echelon form.

## Parameters

|        |                                                        |
|--------|--------------------------------------------------------|
| $Fi$   | base Field                                             |
| $Uplo$ | chosse if the procedure is based on row or column      |
| $N$    | size of the matrix                                     |
| $s$    | order of qausiseparability                             |
| $r$    | rank                                                   |
| $A$    | the matrix that will sotck the expanded representation |
| $lda$  | leading dimension of A                                 |
| $X$    | matrix to expand                                       |
| $ldx$  | leading dimension of X                                 |
| $K$    | stock the position of the blocks in A                  |
| $M$    | permutation matrix                                     |
| $T$    | stock the operation done in the procedure              |

**11.21.3.85 Bruhat2EchelonPermutation()**

```
void Bruhat2EchelonPermutation (
 size_t N,
 size_t R,
 const size_t * P,
 const size_t * Q,
 size_t * M) [inline]
```

Bruhat2EchelonPermutation (N,R,P,Q) Compute M such that LM or MU is in echelon form where L or U are factors of the Bruhat Rpresentation.

#### Parameters

|     |     |                           |
|-----|-----|---------------------------|
| in  | $N$ | size of the matrix        |
| in  | $R$ | rank                      |
| in  | $P$ | permutation Matrix        |
| in  | $Q$ | permutation Matrix        |
| out | $M$ | output permutation matrix |

#### 11.21.3.86 TInverter() [1/2]

```
size_t * TInverter (
 size_t * T,
 size_t r)
```

#### 11.21.3.87 ComputeRPermutation() [1/2]

```
template<class Field >
void ComputeRPermutation (
 const Field & Fi,
 size_t N,
 size_t r,
 const size_t * P,
 const size_t * Q,
 size_t * R,
 size_t * MU,
 size_t * ML)
```

#### 11.21.3.88 productBruhatxTS() [1/2]

```
template<class Field >
void productBruhatxTS (
 const Field & Fi,
 size_t N,
 size_t s,
 size_t r,
 const size_t * P,
 const size_t * Q,
 const typename Field::Element_ptr Xu,
 size_t ldu,
 size_t NbBlocksU,
 size_t * Ku,
 size_t * Tu,
 size_t * MU,
 const typename Field::Element_ptr Xl,
 size_t ldl,
 size_t NbBlocksL,
 size_t * Kl,
 size_t * Tl,
```

```

 size_t * ML,
 typename Field::Element_ptr B,
 size_t t,
 size_t ldb,
 typename Field::Element_ptr C,
 size_t ldc)

```

productBruhatxTS Compute the product between the CRE compact representation of a matrix A and B a tall matrix

### 11.21.3.89 LQUPtoInverseOfFullRankMinor() [1/2]

```

template<class Field >
Field::Element_ptr LQUPtoInverseOfFullRankMinor (
 const Field & F,
 const size_t rank,
 typename Field::Element_ptr A_factors,
 const size_t lda,
 const size_t * QtPointer,
 typename Field::Element_ptr X,
 const size_t idx)

```

LQUPtoInverseOfFullRankMinor.

Suppose A has been factorized as L.Q.U.P, with rank r. Then Qt.A.Pt has an invertible leading principal r x r submatrix This procedure efficiently computes the inverse of this minor and puts it into X.

#### Note

It changes the lower entries of A\_factors in the process (NB: unless A was nonsingular and square)

#### Parameters

|                  |                                                            |
|------------------|------------------------------------------------------------|
| <i>F</i>         | base field                                                 |
| <i>rank</i>      | rank of the matrix.                                        |
| <i>A_factors</i> | matrix containing the L and U entries of the factorization |
| <i>lda</i>       | leading dimension of A                                     |
| <i>QtPointer</i> | theLQUP->getQ()->getPointer() (note: getQ returns Qt!)     |
| <i>X</i>         | desired location for output                                |
| <i>idx</i>       | leading dimension of X                                     |

### 11.21.3.90 RandomNullSpaceVector() [2/3]

```

template<class Field >
void RandomNullSpaceVector (
 const Field & F,
 const FFLAS::FFLAS_SIDE Side,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,

```

```
typename Field::Element_ptr X,
const size_t incX)
```

Solve  $LX = B$  or  $XL = B$  in place.

L is  $M \times M$  if Side == [FFLAS::FflasLeft](#) and  $N \times N$  if Side == [FFLAS::FflasRight](#), B is  $M \times N$ . Only the R non trivial column of L are stored in the  $M \times R$  matrix L Requirement : so that L could be expanded in-place Computes a vector of the Left/Right nullspace of the matrix A.

#### Parameters

|         |             |                                                                                  |
|---------|-------------|----------------------------------------------------------------------------------|
|         | <i>F</i>    | The computation domain                                                           |
|         | <i>Side</i> | decides whether it computes the left (FflasLeft) or right (FflasRight) nullspace |
|         | <i>M</i>    | number of rows                                                                   |
|         | <i>N</i>    | number of columns                                                                |
| in, out | <i>A</i>    | input matrix of dimension $M \times N$ , A is modified to its LU version         |
|         | <i>lda</i>  | leading dimension of A                                                           |
| out     | <i>X</i>    | output vector                                                                    |
|         | <i>incX</i> | increment of X                                                                   |

#### 11.21.3.91 solveLB() [1/2]

```
template<class Field >
void solveLB (
 const Field & F,
 const FFLAS::FFLAS_SIDE Side,
 const size_t M,
 const size_t N,
 const size_t R,
 typename Field::Element_ptr L,
 const size_t ldl,
 const size_t * Q,
 typename Field::Element_ptr B,
 const size_t ldb)
```

#### 11.21.3.92 solveLB2() [1/2]

```
template<class Field >
void solveLB2 (
 const Field & F,
 const FFLAS::FFLAS_SIDE Side,
 const size_t M,
 const size_t N,
 const size_t R,
 typename Field::Element_ptr L,
 const size_t ldl,
 const size_t * Q,
 typename Field::Element_ptr B,
 const size_t ldb)
```

**11.21.3.93 TInverter()** [2/2]

```
size_t * TInverter (
 const size_t * T,
 size_t r) [inline]
```

**11.21.3.94 ComputeRPermutation()** [2/2]

```
template<class Field >
void ComputeRPermutation (
 const Field & Fi,
 size_t N,
 size_t r,
 const size_t * P,
 const size_t * Q,
 size_t * R,
 const size_t * MU,
 const size_t * ML) [inline]
```

**11.21.3.95 expandLCRE()**

```
template<class Field >
Field::Element_ptr expandLCRE (
 const Field & Fi,
 size_t N,
 size_t s,
 size_t r,
 size_t * R,
 size_t i,
 typename Field::ConstElement_ptr Xu,
 size_t ldu,
 size_t NbBlocksU,
 const size_t * Ku,
 const size_t * Tuinv,
 typename Field::ConstElement_ptr Xl,
 size_t ldl,
 size_t NbBlocksL,
 const size_t * Kl,
 const size_t * Tlinv,
 typename Field::Element_ptr CRE,
 size_t ldcre) [inline]
```

Expands an anti-diagonal block of a left triangular matrix from its compact Bruhat representation.

**11.21.3.96 productBruhatxTS()** [2/2]

```
template<class Field >
void productBruhatxTS (
 const Field & Fi,
 size_t N,
 size_t s,
 size_t r,
```



```

size_t t,
const size_t * P,
const size_t * Q,
typename Field::ConstElement_ptr Xu,
size_t ldu,
size_t NbBlocksU,
const size_t * Ku,
const size_t * Tu,
const size_t * MU,
typename Field::ConstElement_ptr Xl,
size_t ldl,
size_t NbBlocksL,
const size_t * Kl,
const size_t * Tl,
const size_t * ML,
typename Field::Element_ptr B,
size_t ldb,
const typename Field::Element beta,
typename Field::Element_ptr D,
size_t ldd) [inline]

```

Compute the product of a left-triangular quasi-separable matrix A, represented by a compact Bruhat generator, with a dense rectangular matrix B:  $C \leftarrow A \times B + \text{beta}C$ .

#### Parameters

|         |                  |                                                                                             |
|---------|------------------|---------------------------------------------------------------------------------------------|
|         | <i>F</i>         | the base field                                                                              |
|         | <i>N</i>         | the order of A                                                                              |
|         | <i>s</i>         | the order of quasiseparability of A                                                         |
|         | <i>r</i>         | the number of pivots in the left-triangular par of the rank profile matrix of A             |
|         | <i>t</i>         | the number of columns of B                                                                  |
|         | <i>P</i>         | the row indices of the pivots of A                                                          |
|         | <i>Q</i>         | the column indices of the pivots of A                                                       |
|         | <i>Xu</i>        | the compact storage of U: Du blocks in the first s rows, Su blocks in the last s rows       |
|         | <i>ldxu</i>      | the leading dimension of Xu                                                                 |
|         | <i>NbBlocksU</i> | the number of diagonal blocks in the compact storage of U                                   |
|         | <i>Ku</i>        | the list of starting column positions for each block of the storage of U                    |
|         | <i>Tu</i>        | the folding matrix for the compact storage of U: $Du + TuSu$ is in row echelon form         |
|         | <i>Mu</i>        | a permutation matrix such that $Mu(Du + TuSu)$ is the U factor of the Bruhat generator      |
|         | <i>Xl</i>        | the compact storage of L: Dl blocks in the first s columns, Sl blocks in the last s columns |
|         | <i>ldxl</i>      | the leading dimension of Xl                                                                 |
|         | <i>NbBlocksL</i> | the number of diagonal blocks in the compact storage of L                                   |
|         | <i>Kl</i>        | the list of starting row positions for each block of the storage of L                       |
|         | <i>Tl</i>        | the folding matrix for the compact storage of L: $Dl + SlTl$ is in column echelon form      |
|         | <i>Ml</i>        | a permutation matrix such that $(Dl + SlTl)Ml$ is the L factor of the Bruhat generator      |
|         | <i>B</i>         | an $N \times t$ dense matrix                                                                |
|         | <i>ldb</i>       | leading dimension of B                                                                      |
|         | <i>beta</i>      | scaling constant                                                                            |
| in, out | <i>C</i>         | output matrix                                                                               |
|         | <i>ldc</i>       | leading dimension of C                                                                      |

**Bibliography** Pernet C. and Storjohann A. *Time and space efficient generators for quasiseparable matrices*, JSC (85), 2018, doi:10.1016/j.jsc.2017.07.010

### 11.21.3.97 Danilevski()

```
template<class Field , class Polynomial >
std::list< Polynomial > & Danilevski (
 const Field & F,
 std::list< Polynomial > & charp,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda)
```

### 11.21.3.98 buildMatrix()

```
template<class Field >
Field::Element_ptr buildMatrix (
 const Field & F,
 typename Field::ConstElement_ptr E,
 typename Field::ConstElement_ptr C,
 const size_t lda,
 const size_t * B,
 const size_t * T,
 const size_t me,
 const size_t mc,
 const size_t lambda,
 const size_t mu)
```

**Bug** is this :

### 11.21.3.99 CharPoly() [4/8]

```
FFPACK::RNSInteger< FFPACK::rns_double >::Element_ptr CharPoly (
 const FFPACK::RNSInteger< FFPACK::rns_double > & F,
 typename FFPACK::RNSInteger< FFPACK::rns_double >::Element_ptr charp,
 const size_t N,
 typename FFPACK::RNSInteger< FFPACK::rns_double >::Element_ptr A,
 const size_t lda,
 Givaro::ZRing< Givaro::Integer >::RandIter & G,
 const FFPACK_CHARPOLY_TAG CharpTag,
 size_t degree) [inline]
```

### 11.21.3.100 CharPoly() [5/8]

```
template<>
Givaro::Poly1Dom< Givaro::ZRing< Givaro::Integer > >::Element & CharPoly (
 const Givaro::Poly1Dom< Givaro::ZRing< Givaro::Integer > > & R,
 Givaro::Poly1Dom< Givaro::ZRing< Givaro::Integer > >::Element & charp,
 const size_t N,
 Givaro::Integer * A,
 const size_t lda,
 Givaro::ZRing< Givaro::Integer >::RandIter & G,
 const FFPACK_CHARPOLY_TAG CharpTag,
 size_t degree) [inline]
```

**11.21.3.101 Det()** [3/6]

```
template<class PSHelper >
FFPACK::RNSInteger< FFPACK::rns_double >::Element_ptr & Det (
 const FFPACK::RNSInteger< FFPACK::rns_double > & F,
 typename FFPACK::RNSInteger< FFPACK::rns_double >::Element_ptr & det,
 const size_t N,
 typename FFPACK::RNSInteger< FFPACK::rns_double >::Element_ptr A,
 const size_t lda,
 const PSHelper & psH) [inline]
```

**11.21.3.102 Det()** [4/6]

```
template<class PSHelper >
Givaro::Integer & Det (
 const Givaro::ZRing< Givaro::Integer > & F,
 Givaro::Integer & det,
 const size_t N,
 Givaro::Integer * A,
 const size_t lda,
 const PSHelper & psH,
 size_t * P,
 size_t * Q) [inline]
```

**11.21.3.103 fsytrf\_BC\_Crout()**

```
template<class Field >
bool fsytrf_BC_Crout (
 const Field & F,
 const FFLAS::FFLAS_UPLO UpLo,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr Dinv,
 const size_t incDinv) [inline]
```

**11.21.3.104 fsytrf\_BC\_RL()**

```
template<class Field >
size_t fsytrf_BC_RL (
 const Field & F,
 const FFLAS::FFLAS_UPLO UpLo,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr Dinv,
 const size_t incDinv) [inline]
```

**11.21.3.105 fsytrf\_UP\_RPM\_BC\_RL()**

```
template<class Field >
size_t fsytrf_UP_RPM_BC_RL (
 const Field & F,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr Dinv,
 const size_t incDinv,
 size_t * P) [inline]
```

**11.21.3.106 fsytrf\_LOW\_RPM\_BC\_Crout()**

```
template<class Field >
size_t fsytrf_LOW_RPM_BC_Crout (
 const Field & F,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr Dinv,
 const size_t incDinv,
 size_t * P) [inline]
```

**11.21.3.107 fsytrf\_UP\_RPM\_BC\_Crout()**

```
template<class Field >
size_t fsytrf_UP_RPM_BC_Crout (
 const Field & F,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr Dinv,
 const size_t incDinv,
 size_t * P) [inline]
```

**11.21.3.108 fsytrf\_UP\_RPM()**

```
template<class Field >
size_t fsytrf_UP_RPM (
 const Field & Fi,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr Dinv,
 const size_t incDinv,
 size_t * P,
 size_t BCThreshold) [inline]
```

MathP <- [ [ I ] x P1 | [ [ L (N1+R2) ] [ P2^T ] ] ] x [ P3^T ] [ ----- | --- ] [ [ Q2^T ]

Changing [ U1 V1 | E1 E21 E22 ] into [ U1 E11 E12 V1 E\* E\* ] [ 0 | L2 \ U2 V21 V22 ] [ U4 V41 0 V42 V43 ] [ 0 | M2 0 0 ] [ U3 0 0 V3 ] [ ----- ] [ 0 0 0 ] [ 0 | H1 H21 H22 ] [ 0 | U3 V3 ] [ 0 | 0 ] where U4 is the 2R2 x 2R2 matrix formed by interleaving U2, L2^T and H1

**11.21.3.109 fsytrf\_nonunit() [2/3]**

```
template<class Field >
bool fsytrf_nonunit (
 const Field & F,
 const FFLAS::FFLAS_UPLO UpLo,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr Dinv,
 const size_t incDinv,
 FFLAS::ParSeqHelper::Sequential seq,
 size_t threshold) [inline]
```

**11.21.3.110 fsytrf\_nonunit() [3/3]**

```
template<class Field , class Cut , class Param >
bool fsytrf_nonunit (
 const Field & F,
 const FFLAS::FFLAS_UPLO UpLo,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr Dinv,
 const size_t incDinv,
 FFLAS::ParSeqHelper::Parallel< Cut, Param > par,
 size_t threshold) [inline]
```

**11.21.3.111 fsytrf\_RPM()**

```
template<class Field >
size_t fsytrf_RPM (
 const Field & F,
 const FFLAS::FFLAS_UPLO UpLo,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t * P,
 size_t threshold) [inline]
```

**11.21.3.112 getTridiagonal()**

```
template<class Field >
void getTridiagonal (
 const Field & F,
 const size_t N,
 const size_t R,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 size_t * P,
 typename Field::Element_ptr T,
 const size_t ldt) [inline]
```

**11.21.3.113 LUdivine\_gauss()** [1/2]

```
template<class Field >
size_t LUdivine_gauss (
 const Field & F,
 const FFLAS::FFLAS_DIAG Diag,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t * P,
 size_t * Q,
 const FFPACK::FFPACK_LU_TAG LuTag) [inline]
```

**11.21.3.114 LUdivine\_small()** [1/2]

```
template<class Field >
size_t LUdivine_small (
 const Field & F,
 const FFLAS::FFLAS_DIAG Diag,
 const FFLAS::FFLAS_TRANSPOSE trans,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t * P,
 size_t * Q,
 const FFPACK::FFPACK_LU_TAG LuTag) [inline]
```

**11.21.3.115 LUdivine()** [2/4]

```
template<class Field >
size_t LUdivine (
 const Field & F,
 const FFLAS::FFLAS_DIAG Diag,
 const FFLAS::FFLAS_TRANSPOSE trans,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t * P,
 size_t * Q,
 const FFPACK::FFPACK_LU_TAG LuTag,
 const size_t cutoff) [inline]
```

**Todo** std::swap ?

**11.21.3.116 LUdivine()** [3/4]

```

template<>
size_t LUdivine (
 const Givaro::Modular< Givaro::Integer > & F,
 const FFLAS::FFLAS_DIAG Diag,
 const FFLAS::FFLAS_TRANSPOSE trans,
 const size_t M,
 const size_t N,
 typename Givaro::Integer * A,
 const size_t lda,
 size_t * P,
 size_t * Q,
 const FFPACK::FFPACK_LU_TAG LuTag,
 const size_t cutoff) [inline]

```

**11.21.3.117 MonotonicCompress()**

```

template<class Field >
void MonotonicCompress (
 const Field & F,
 const FFLAS::FFLAS_SIDE Side,
 const size_t M,
 typename Field::Element_ptr A,
 const size_t lda,
 const size_t incA,
 const size_t * MathP,
 const size_t R,
 const size_t maxpiv,
 const size_t rowstomove,
 const std::vector< bool > & ispiv) [inline]

```

**11.21.3.118 MonotonicCompressMorePivots()**

```

template<class Field >
void MonotonicCompressMorePivots (
 const Field & F,
 const FFLAS::FFLAS_SIDE Side,
 const size_t M,
 typename Field::Element_ptr A,
 const size_t lda,
 const size_t incA,
 const size_t * MathP,
 const size_t R,
 const size_t rowstomove,
 const size_t lenP) [inline]

```

**11.21.3.119 MonotonicCompressCycles()**

```

template<class Field >
void MonotonicCompressCycles (
 const Field & F,

```

```

 const FFLAS::FFLAS_SIDE Side,
 const size_t M,
 typename Field::Element_ptr A,
 const size_t lda,
 const size_t incA,
 const size_t * MathP,
 const size_t lenP) [inline]

```

#### 11.21.3.120 MonotonicExpand()

```

template<class Field >
void MonotonicExpand (
 const Field & F,
 const FFLAS::FFLAS_SIDE Side,
 const size_t M,
 typename Field::Element_ptr A,
 const size_t lda,
 const size_t incA,
 const size_t * MathP,
 const size_t R,
 const size_t maxpiv,
 const size_t rowstomove,
 const std::vector< bool > & ispiv)

```

#### 11.21.3.121 applyP\_block()

```

template<class Field >
void applyP_block (
 const Field & F,
 const FFLAS::FFLAS_SIDE Side,
 const FFLAS::FFLAS_TRANSPOSE Trans,
 const size_t M,
 const size_t ibeg,
 const size_t iend,
 typename Field::Element_ptr A,
 const size_t lda,
 const size_t * P) [inline]

```

#### 11.21.3.122 doApplyS()

```

template<class Field >
void doApplyS (
 const Field & F,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr tmp,
 const size_t width,
 const size_t M2,
 const size_t R1,
 const size_t R2,
 const size_t R3,
 const size_t R4) [inline]

```



**11.21.3.123 MatrixApplyS()** [1/3]

```
template<class Field >
void MatrixApplyS (
 const Field & F,
 typename Field::Element_ptr A,
 const size_t lda,
 const size_t width,
 const size_t M2,
 const size_t R1,
 const size_t R2,
 const size_t R3,
 const size_t R4) [inline]
```

**11.21.3.124 MatrixApplyS()** [2/3]

```
template<class Field >
void MatrixApplyS (
 const Field & F,
 typename Field::Element_ptr A,
 const size_t lda,
 const size_t width,
 const size_t M2,
 const size_t R1,
 const size_t R2,
 const size_t R3,
 const size_t R4,
 const FFLAS::ParSeqHelper::Sequential seq) [inline]
```

**11.21.3.125 MatrixApplyS()** [3/3]

```
template<class Field , class Cut , class Param >
void MatrixApplyS (
 const Field & F,
 typename Field::Element_ptr A,
 const size_t lda,
 const size_t width,
 const size_t M2,
 const size_t R1,
 const size_t R2,
 const size_t R3,
 const size_t R4,
 const FFLAS::ParSeqHelper::Parallel< Cut, Param > par) [inline]
```

**11.21.3.126 PermApplyS()**

```
template<class T >
void PermApplyS (
 T * A,
 const size_t lda,
 const size_t width,
 const size_t M2,
 const size_t R1,
 const size_t R2,
 const size_t R3,
 const size_t R4) [inline]
```

**11.21.3.127 doApplyT()**

```
template<class Field >
void doApplyT (
 const Field & F,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr tmp,
 const size_t width,
 const size_t N2,
 const size_t R1,
 const size_t R2,
 const size_t R3,
 const size_t R4) [inline]
```

**11.21.3.128 MatrixApplyT() [1/3]**

```
template<class Field >
void MatrixApplyT (
 const Field & F,
 typename Field::Element_ptr A,
 const size_t lda,
 const size_t width,
 const size_t N2,
 const size_t R1,
 const size_t R2,
 const size_t R3,
 const size_t R4) [inline]
```

**11.21.3.129 MatrixApplyT() [2/3]**

```
template<class Field >
void MatrixApplyT (
 const Field & F,
 typename Field::Element_ptr A,
 const size_t lda,
 const size_t width,
 const size_t N2,
 const size_t R1,
 const size_t R2,
 const size_t R3,
 const size_t R4,
 const FFLAS::ParSeqHelper::Sequential seq) [inline]
```

**11.21.3.130 MatrixApplyT() [3/3]**

```
template<class Field , class Cut , class Param >
void MatrixApplyT (
 const Field & F,
 typename Field::Element_ptr A,
 const size_t lda,
 const size_t width,
```

```

const size_t N2,
const size_t R1,
const size_t R2,
const size_t R3,
const size_t R4,
const FFLAS::ParSeqHelper::Parallel< Cut, Param > par) [inline]

```

### 11.21.3.131 PermApplyT()

```

template<class T >
void PermApplyT (
 T * A,
 const size_t lda,
 const size_t width,
 const size_t N2,
 const size_t R1,
 const size_t R2,
 const size_t R3,
 const size_t R4) [inline]

```

### 11.21.3.132 composePermutationsLLL()

```

void composePermutationsLLL (
 size_t * P1,
 const size_t * P2,
 const size_t R,
 const size_t N) [inline]

```

Computes  $P1 \times \text{Diag}(I_R, P2)$  where  $P1$  is a LAPACK and  $P2$  a LAPACK permutation and store the result in  $P1$  as a LAPACK permutation.

#### Parameters

|         |      |                                  |
|---------|------|----------------------------------|
| in, out | $P1$ | a LAPACK permutation of size N   |
|         | $P2$ | a LAPACK permutation of size N-R |

### 11.21.3.133 composePermutationsLLM()

```

void composePermutationsLLM (
 size_t * MathP,
 const size_t * P1,
 const size_t * P2,
 const size_t R,
 const size_t N) [inline]

```

Computes  $P1 \times \text{Diag}(I_R, P2)$  where  $P1$  is a LAPACK and  $P2$  a LAPACK permutation and store the result in  $\text{MathP}$  as a MathPermutation format.

#### Parameters

|     |  |  |
|-----|--|--|
| out |  |  |
|-----|--|--|

a MathPermutation of size N

#### Parameters

|           |                                  |
|-----------|----------------------------------|
| <i>P1</i> | a LAPACK permutation of size N   |
| <i>P2</i> | a LAPACK permutation of size N-R |

### 11.21.3.134 composePermutationsMLM()

```
void composePermutationsMLM (
 size_t * MathP1,
 const size_t * P2,
 const size_t R,
 const size_t N) [inline]
```

Computes MathP1 x Diag (I\_R, P2) where MathP1 is a MathPermutation and P2 a LAPACK permutation and store the result in MathP1 as a MathPermutation format.

#### Parameters

|         |               |                                  |
|---------|---------------|----------------------------------|
| in, out | <i>MathP1</i> | a MathPermutation of size N      |
|         | <i>P2</i>     | a LAPACK permutation of size N-R |

### 11.21.3.135 cyclic\_shift\_mathPerm()

```
void cyclic_shift_mathPerm (
 size_t * P,
 const size_t s) [inline]
```

### 11.21.3.136 cyclic\_shift\_row\_col() [1/2]

```
template<class Field >
void cyclic_shift_row_col (
 const Field & F,
 typename Field::Element_ptr A,
 size_t m,
 size_t n,
 size_t lda) [inline]
```

### 11.21.3.137 cyclic\_shift\_row() [1/3]

```
template<class Field >
void cyclic_shift_row (
 const Field & F,
 typename Field::Element_ptr A,
 size_t m,
 size_t n,
 size_t lda) [inline]
```

**11.21.3.138 cyclic\_shift\_row()** [2/3]

```
template<typename T >
void cyclic_shift_row (
 const RNSIntegerMod< T > & F,
 typename T::Element_ptr A,
 size_t m,
 size_t n,
 size_t lda) [inline]
```

**11.21.3.139 cyclic\_shift\_col()** [1/3]

```
template<class Field >
void cyclic_shift_col (
 const Field & F,
 typename Field::Element_ptr A,
 size_t m,
 size_t n,
 size_t lda) [inline]
```

**11.21.3.140 cyclic\_shift\_col()** [2/3]

```
template<typename T >
void cyclic_shift_col (
 const RNSIntegerMod< T > & F,
 typename T::Element_ptr A,
 size_t m,
 size_t n,
 size_t lda) [inline]
```

**11.21.3.141 PLUQ\_basecaseV3()**

```
template<class Field >
size_t PLUQ_basecaseV3 (
 const Field & Fi,
 const FFLAS::FFLAS_DIAG Diag,
 const size_t M,
 const size_t N,
 typename Field::Element * A,
 const size_t lda,
 size_t * P,
 size_t * Q) [inline]
```

**11.21.3.142 PLUQ\_basecaseV2()**

```
template<class Field >
size_t PLUQ_basecaseV2 (
 const Field & Fi,
 const FFLAS::FFLAS_DIAG Diag,
 const size_t M,
 const size_t N,
 typename Field::Element * A,
 const size_t lda,
 size_t * P,
 size_t * Q) [inline]
```

**11.21.3.143 PLUQ\_basecaseCrout()**

```
template<class Field >
size_t PLUQ_basecaseCrout (
 const Field & Fi,
 const FFLAS::FFLAS_DIAG Diag,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t * P,
 size_t * Q) [inline]
```

**11.21.3.144 \_PLUQ()**

```
template<class Field >
size_t _PLUQ (
 const Field & Fi,
 const FFLAS::FFLAS_DIAG Diag,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t * P,
 size_t * Q,
 size_t BCThreshold) [inline]
```

**11.21.3.145 PLUQ() [4/6]**

```
template<class Cut , class Param >
size_t PLUQ (
 const Givaro::Modular< Givaro::Integer > & F,
 const FFLAS::FFLAS_DIAG Diag,
 const size_t M,
 const size_t N,
 typename Givaro::Integer * A,
 const size_t lda,
 size_t * P,
 size_t * Q,
 size_t BCThreshold,
 FFLAS::ParSeqHelper::Parallel< Cut, Param > & PSHelper) [inline]
```

**11.21.3.146 threads\_fgemm()**

```
template<class Field >
void threads_fgemm (
 const size_t m,
 const size_t n,
 const size_t r,
 int nbthreads,
 size_t * W1,
 size_t * W2,
 size_t * W3,
 size_t gamma)
```

**11.21.3.147 threads\_ftrsm()**

```
template<class Field >
void threads_ftrsm (
 const size_t m,
 const size_t n,
 int nbthreads,
 size_t * t1,
 size_t * t2)
```

**11.21.3.148 PLUQ() [5/6]**

```
template<class Field >
size_t PLUQ (
 const Field & Fi,
 const FFLAS::FFLAS_DIAG Diag,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t * P,
 size_t * Q,
 const FFLAS::ParSeqHelper::Parallel< FFLAS::CuttingStrategy::Recursive, FFLAS::StrategyParameter
> & PSHelper) [inline]
```

**11.21.3.149 fflas\_const\_cast() [1/3]**

```
template<>
rns_double_elt_ptr fflas_const_cast (
 rns_double_elt_cstptr x) [inline]
```

**11.21.3.150 fflas\_const\_cast() [2/3]**

```
template<>
rns_double_elt_cstptr fflas_const_cast (
 rns_double_elt_ptr x) [inline]
```

**11.21.3.151 cyclic\_shift\_row\_col() [2/2]**

```
template<typename Base_t >
void cyclic_shift_row_col (
 Base_t * A,
 size_t m,
 size_t n,
 size_t lda)
```

**11.21.3.152 cyclic\_shift\_row()** [3/3]

```
template INST_OR_DECL void cyclic_shift_row (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 FFLAS_ELT * A,
 size_t m,
 size_t n,
 size_t lda)
```

**11.21.3.153 cyclic\_shift\_col()** [3/3]

```
template INST_OR_DECL void cyclic_shift_col (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 FFLAS_ELT * A,
 size_t m,
 size_t n,
 size_t lda)
```

**11.21.3.154 applyP()** [4/4]

```
template INST_OR_DECL void applyP (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const FFLAS::FFLAS_SIDE Side,
 const FFLAS::FFLAS_TRANSPOSE Trans,
 const size_t M,
 const size_t ibeg,
 const size_t iend,
 FFLAS_ELT * A,
 const size_t lda,
 const size_t * P)
```

**11.21.3.155 fgetrs()** [3/4]

```
template INST_OR_DECL void fgetrs (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const FFLAS::FFLAS_SIDE Side,
 const size_t M,
 const size_t N,
 const size_t R,
 FFLAS_ELT * A,
 const size_t lda,
 const size_t * P,
 const size_t * Q,
 FFLAS_ELT * B,
 const size_t ldb,
 int * info)
```



**11.21.3.156 fgetrs()** [4/4]

```

template INST_OR_DECL FFLAS_ELT * fgetrs (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const FFLAS::FFLAS_SIDE Side,
 const size_t M,
 const size_t N,
 const size_t NRHS,
 const size_t R,
 FFLAS_ELT * A,
 const size_t lda,
 const size_t * P,
 const size_t * Q,
 FFLAS_ELT * X,
 const size_t ldX,
 const FFLAS_ELT * B,
 const size_t ldb,
 int * info)

```

**11.21.3.157 fgesv()** [3/4]

```

template INST_OR_DECL size_t fgesv (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const FFLAS::FFLAS_SIDE Side,
 const size_t M,
 const size_t N,
 FFLAS_ELT * A,
 const size_t lda,
 FFLAS_ELT * B,
 const size_t ldb,
 int * info)

```

**11.21.3.158 fgesv()** [4/4]

```

template INST_OR_DECL size_t fgesv (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const FFLAS::FFLAS_SIDE Side,
 const size_t M,
 const size_t N,
 const size_t NRHS,
 FFLAS_ELT * A,
 const size_t lda,
 FFLAS_ELT * X,
 const size_t ldX,
 const FFLAS_ELT * B,
 const size_t ldb,
 int * info)

```

**11.21.3.159 ftrtri()** [2/2]

```

template INST_OR_DECL void ftrtri (
 const FFLAS_FIELD< FFLAS_ELT > & F,

```

```

const FFLAS::FFLAS_UPLO Uplo,
const FFLAS::FFLAS_DIAG Diag,
const size_t N,
FFLAS_ELT * A,
const size_t lda,
const size_t threshold)

```

#### 11.21.3.160 trinv\_left() [2/2]

```

template INST_OR_DECL void trinv_left (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t N,
 const FFLAS_ELT * L,
 const size_t ldl,
 FFLAS_ELT * X,
 const size_t idx)

```

#### 11.21.3.161 ftrtrm() [2/2]

```

template INST_OR_DECL void ftrtrm (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const FFLAS::FFLAS_SIDE side,
 const FFLAS::FFLAS_DIAG diag,
 const size_t N,
 FFLAS_ELT * A,
 const size_t lda)

```

#### 11.21.3.162 PLUQ() [6/6]

```

template INST_OR_DECL size_t PLUQ (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const FFLAS::FFLAS_DIAG Diag,
 const size_t M,
 const size_t N,
 FFLAS_ELT * A,
 const size_t lda,
 size_t * P,
 size_t * Q)

```

#### 11.21.3.163 LUdivine() [4/4]

```

template INST_OR_DECL size_t LUdivine (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const FFLAS::FFLAS_DIAG Diag,
 const FFLAS::FFLAS_TRANSPOSE trans,
 const size_t M,
 const size_t N,
 FFLAS_ELT * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const FFPACK_LU_TAG LuTag,
 const size_t cutoff)

```

**11.21.3.164 LUdivine\_small()** [2/2]

```
template INST_OR_DECL size_t LUdivine_small (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const FFLAS::FFLAS_DIAG Diag,
 const FFLAS::FFLAS_TRANSPOSE trans,
 const size_t M,
 const size_t N,
 FFLAS_ELT * A,
 const size_t lda,
 size_t * P,
 size_t * Q,
 const FFPACK_LU_TAG LuTag)
```

**11.21.3.165 LUdivine\_gauss()** [2/2]

```
template INST_OR_DECL size_t LUdivine_gauss (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const FFLAS::FFLAS_DIAG Diag,
 const size_t M,
 const size_t N,
 FFLAS_ELT * A,
 const size_t lda,
 size_t * P,
 size_t * Q,
 const FFPACK_LU_TAG LuTag)
```

**11.21.3.166 RowEchelonForm()** [3/3]

```
template INST_OR_DECL size_t RowEchelonForm (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t M,
 const size_t N,
 FFLAS_ELT * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const FFPACK_LU_TAG LuTag)
```

**11.21.3.167 ReducedRowEchelonForm()** [3/3]

```
template INST_OR_DECL size_t ReducedRowEchelonForm (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t M,
 const size_t N,
 FFLAS_ELT * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const FFPACK_LU_TAG LuTag)
```

**11.21.3.168 ColumnEchelonForm()** [3/3]

```
template INST_OR_DECL size_t ColumnEchelonForm (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t M,
 const size_t N,
 FFLAS_ELT * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const FFPACK_LU_TAG LuTag)
```

**11.21.3.169 ReducedColumnEchelonForm()** [3/3]

```
template INST_OR_DECL size_t ReducedColumnEchelonForm (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t M,
 const size_t N,
 FFLAS_ELT * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const FFPACK_LU_TAG LuTag)
```

**11.21.3.170 Invert()** [3/4]

```
template INST_OR_DECL FFLAS_ELT * Invert (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t M,
 FFLAS_ELT * A,
 const size_t lda,
 int & nullity)
```

**11.21.3.171 Invert()** [4/4]

```
template INST_OR_DECL FFLAS_ELT * Invert (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t M,
 const FFLAS_ELT * A,
 const size_t lda,
 FFLAS_ELT * X,
 const size_t ldx,
 int & nullity)
```

**11.21.3.172 Invert2()** [2/2]

```
template INST_OR_DECL FFLAS_ELT * Invert2 (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t M,
 FFLAS_ELT * A,
 const size_t lda,
 FFLAS_ELT * X,
 const size_t ldx,
 int & nullity)
```

**11.21.3.173 CharPoly()** [6/8]

```
template INST_OR_DECL std::list< Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > >::Element > &
CharPoly (
 const Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > > & R,
 std::list< Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > >::Element > & charp,
 const size_t N,
 FFLAS_ELT * A,
 const size_t lda,
 FFLAS_FIELD< FFLAS_ELT >::RandIter & G,
 const FFPACK_CHARPOLY_TAG CharpTag,
 const size_t degree)
```

**11.21.3.174 CharPoly()** [7/8]

```
template INST_OR_DECL Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > >::Element & CharPoly (
 const Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > > & R,
 Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > >::Element & charp,
 const size_t N,
 FFLAS_ELT * A,
 const size_t lda,
 FFLAS_FIELD< FFLAS_ELT >::RandIter & G,
 const FFPACK_CHARPOLY_TAG CharpTag,
 const size_t degree)
```

**11.21.3.175 CharPoly()** [8/8]

```
template INST_OR_DECL Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > >::Element & CharPoly (
 const Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > > & R,
 Givaro::Poly1Dom< FFLAS_FIELD< FFLAS_ELT > >::Element & charp,
 const size_t N,
 FFLAS_ELT * A,
 const size_t lda,
 const FFPACK_CHARPOLY_TAG CharpTag,
 const size_t degree)
```

**11.21.3.176 MinPoly()** [3/4]

```
template INST_OR_DECL std::vector< FFLAS_ELT > & MinPoly (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 std::vector< FFLAS_ELT > & minP,
 const size_t N,
 const FFLAS_ELT * A,
 const size_t lda,
 FFLAS_FIELD< FFLAS_ELT >::RandIter & G)
```

**11.21.3.177 MinPoly()** [4/4]

```
template INST_OR_DECL std::vector< FFLAS_ELT > & MinPoly (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 std::vector< FFLAS_ELT > & minP,
 const size_t N,
 const FFLAS_ELT * A,
 const size_t lda)
```

**11.21.3.178 MatVecMinPoly()** [2/2]

```
template INST_OR_DECL std::vector< FFLAS_ELT > & MatVecMinPoly (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 std::vector< FFLAS_ELT > & minP,
 const size_t N,
 const FFLAS_ELT * A,
 const size_t lda,
 const FFLAS_ELT * V,
 const size_t incv)
```

**11.21.3.179 KrylovElim()**

```
template INST_OR_DECL size_t KrylovElim (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t M,
 const size_t N,
 FFLAS_ELT * A,
 const size_t lda,
 size_t * P,
 size_t * Q,
 const size_t deg,
 size_t * iterates,
 size_t * inviterates,
 const size_t maxit,
 size_t virt)
```

**11.21.3.180 SpecRankProfile()**

```
template INST_OR_DECL size_t SpecRankProfile (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t M,
 const size_t N,
 FFLAS_ELT * A,
 const size_t lda,
 const size_t deg,
 size_t * rankProfile)
```

**11.21.3.181 Rank()** [3/3]

```
template INST_OR_DECL size_t Rank (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t M,
 const size_t N,
 FFLAS_ELT * A,
 const size_t lda)
```

**11.21.3.182 IsSingular()** [2/2]

```
template INST_OR_DECL bool IsSingular (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t M,
 const size_t N,
 FFLAS_ELT * A,
 const size_t lda)
```

**11.21.3.183 Det()** [5/6]

```
template INST_OR_DECL FFLAS_ELT & Det (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 FFLAS_ELT & det,
 const size_t N,
 FFLAS_ELT * A,
 const size_t lda,
 size_t * P,
 size_t * Q)
```

**11.21.3.184 Det()** [6/6]

```
template INST_OR_DECL FFLAS_ELT & Det (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 FFLAS_ELT & det,
 const size_t N,
 FFLAS_ELT * A,
 const size_t lda,
 const FFLAS::ParSeqHelper::Parallel< FFLAS::CuttingStrategy::Recursive, FFLAS::StrategyParameter
> & parH,
 size_t * P,
 size_t * Q)
```

**11.21.3.185 Solve()** [3/3]

```
template INST_OR_DECL FFLAS_ELT * Solve (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t M,
 FFLAS_ELT * A,
 const size_t lda,
 FFLAS_ELT * x,
 const int incx,
 const FFLAS_ELT * b,
 const int incb)
```

**11.21.3.186 solveLB()** [2/2]

```
template INST_OR_DECL void solveLB (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const FFLAS::FFLAS_SIDE Side,
 const size_t M,
 const size_t N,
 const size_t R,
 FFLAS_ELT * L,
 const size_t ldl,
 const size_t * Q,
 FFLAS_ELT * B,
 const size_t ldb)
```

**11.21.3.187 solveLB2()** [2/2]

```
template INST_OR_DECL void solveLB2 (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const FFLAS::FFLAS_SIDE Side,
 const size_t M,
 const size_t N,
 const size_t R,
 FFLAS_ELT * L,
 const size_t ldl,
 const size_t * Q,
 FFLAS_ELT * B,
 const size_t ldb)
```

**11.21.3.188 RandomNullSpaceVector()** [3/3]

```
template INST_OR_DECL void RandomNullSpaceVector (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const FFLAS::FFLAS_SIDE Side,
 const size_t M,
 const size_t N,
 FFLAS_ELT * A,
 const size_t lda,
 FFLAS_ELT * X,
 const size_t incX)
```

**11.21.3.189 NullSpaceBasis()** [2/2]

```
template INST_OR_DECL size_t NullSpaceBasis (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const FFLAS::FFLAS_SIDE Side,
 const size_t M,
 const size_t N,
 FFLAS_ELT * A,
 const size_t lda,
 FFLAS_ELT *& NS,
 size_t & ldn,
 size_t & NSdim)
```

**11.21.3.190 RowRankProfile()** [3/3]

```
template INST_OR_DECL size_t RowRankProfile (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t M,
 const size_t N,
 FFLAS_ELT * A,
 const size_t lda,
 size_t *& rkprofile,
 const FFPACK_LU_TAG LuTag)
```



**11.21.3.191 ColumnRankProfile()** [3/3]

```
template INST_OR_DECL size_t ColumnRankProfile (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t M,
 const size_t N,
 FFLAS_ELT * A,
 const size_t lda,
 size_t *& rkprofile,
 const FFPACK_LU_TAG LuTag)
```

**11.21.3.192 RowRankProfileSubmatrixIndices()** [2/2]

```
template INST_OR_DECL size_t RowRankProfileSubmatrixIndices (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t M,
 const size_t N,
 FFLAS_ELT * A,
 const size_t lda,
 size_t *& rowindices,
 size_t *& colindices,
 size_t & R)
```

**11.21.3.193 ColRankProfileSubmatrixIndices()** [2/2]

```
template INST_OR_DECL size_t ColRankProfileSubmatrixIndices (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t M,
 const size_t N,
 FFLAS_ELT * A,
 const size_t lda,
 size_t *& rowindices,
 size_t *& colindices,
 size_t & R)
```

**11.21.3.194 RowRankProfileSubmatrix()** [2/2]

```
template INST_OR_DECL size_t RowRankProfileSubmatrix (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t M,
 const size_t N,
 FFLAS_ELT * A,
 const size_t lda,
 FFLAS_ELT *& X,
 size_t & R)
```

**11.21.3.195 ColRankProfileSubmatrix()** [2/2]

```
template INST_OR_DECL size_t ColRankProfileSubmatrix (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t M,
 const size_t N,
 FFLAS_ELT * A,
 const size_t lda,
 FFLAS_ELT *& X,
 size_t & R)
```

**11.21.3.196** `getTriangular< FFLAS_FIELD< FFLAS_ELT > >()` [1/2]

```
template INST_OR_DECL void getTriangular< FFLAS_FIELD< FFLAS_ELT > > > (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const FFLAS::FFLAS_UPLO Uplo,
 const FFLAS::FFLAS_DIAG diag,
 const size_t M,
 const size_t N,
 const size_t R,
 const FFLAS_ELT * A,
 const size_t lda,
 FFLAS_ELT * T,
 const size_t ldt,
 const bool OnlyNonZeroVectors)
```

**11.21.3.197** `getTriangular< FFLAS_FIELD< FFLAS_ELT > >()` [2/2]

```
template INST_OR_DECL void getTriangular< FFLAS_FIELD< FFLAS_ELT > > > (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const FFLAS::FFLAS_UPLO Uplo,
 const FFLAS::FFLAS_DIAG diag,
 const size_t M,
 const size_t N,
 const size_t R,
 FFLAS_ELT * A,
 const size_t lda)
```

**11.21.3.198** `getEchelonForm< FFLAS_FIELD< FFLAS_ELT > >()` [1/2]

```
template INST_OR_DECL void getEchelonForm< FFLAS_FIELD< FFLAS_ELT > > > (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const FFLAS::FFLAS_UPLO Uplo,
 const FFLAS::FFLAS_DIAG diag,
 const size_t M,
 const size_t N,
 const size_t R,
 const size_t * P,
 const FFLAS_ELT * A,
 const size_t lda,
 FFLAS_ELT * T,
 const size_t ldt,
 const bool OnlyNonZeroVectors,
 const FFPACK_LU_TAG LuTag)
```

**11.21.3.199** `getEchelonForm< FFLAS_FIELD< FFLAS_ELT > >()` [2/2]

```
template INST_OR_DECL void getEchelonForm< FFLAS_FIELD< FFLAS_ELT > > > (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const FFLAS::FFLAS_UPLO Uplo,
 const FFLAS::FFLAS_DIAG diag,
 const size_t M,
 const size_t N,
 const size_t R,
 const size_t * P,
 FFLAS_ELT * A,
 const size_t lda,
 const FFPACK_LU_TAG LuTag)
```

**11.21.3.200 getEchelonTransform< FFLAS\_FIELD< FFLAS\_ELT > >()**

```
template INST_OR_DECL void getEchelonTransform< FFLAS_FIELD< FFLAS_ELT > > > (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const FFLAS::FFLAS_UPLO Uplo,
 const FFLAS::FFLAS_DIAG diag,
 const size_t M,
 const size_t N,
 const size_t R,
 const size_t * P,
 const size_t * Q,
 const FFLAS_ELT * A,
 const size_t lda,
 FFLAS_ELT * T,
 const size_t ldt,
 const FFPACK_LU_TAG LuTag)
```

**11.21.3.201 getReducedEchelonForm< FFLAS\_FIELD< FFLAS\_ELT > >() [1/2]**

```
template INST_OR_DECL void getReducedEchelonForm< FFLAS_FIELD< FFLAS_ELT > > > (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const FFLAS::FFLAS_UPLO Uplo,
 const size_t M,
 const size_t N,
 const size_t R,
 const size_t * P,
 const FFLAS_ELT * A,
 const size_t lda,
 FFLAS_ELT * T,
 const size_t ldt,
 const bool OnlyNonZeroVectors,
 const FFPACK_LU_TAG LuTag)
```

**11.21.3.202 getReducedEchelonForm< FFLAS\_FIELD< FFLAS\_ELT > >() [2/2]**

```
template INST_OR_DECL void getReducedEchelonForm< FFLAS_FIELD< FFLAS_ELT > > > (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const FFLAS::FFLAS_UPLO Uplo,
 const size_t M,
 const size_t N,
 const size_t R,
 const size_t * P,
 FFLAS_ELT * A,
 const size_t lda,
 const FFPACK_LU_TAG LuTag)
```

**11.21.3.203 getReducedEchelonTransform< FFLAS\_FIELD< FFLAS\_ELT > >()**

```
template INST_OR_DECL void getReducedEchelonTransform< FFLAS_FIELD< FFLAS_ELT > > > (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const FFLAS::FFLAS_UPLO Uplo,
 const size_t M,
```

```

 const size_t N,
 const size_t R,
 const size_t * P,
 const size_t * Q,
 const FFLAS_ELT * A,
 const size_t lda,
 FFLAS_ELT * T,
 const size_t ldt,
 const FFPACK_LU_TAG LuTag)

```

#### 11.21.3.204 LQUPtoInverseOfFullRankMinor() [2/2]

```

template INST_OR_DECL FFLAS_ELT * LQUPtoInverseOfFullRankMinor (
 const FFLAS_FIELD< FFLAS_ELT > & F,
 const size_t rank,
 FFLAS_ELT * A_factors,
 const size_t lda,
 const size_t * QtPointer,
 FFLAS_ELT * X,
 const size_t ldx)

```

#### 11.21.3.205 fflas\_const\_cast() [3/3]

```

template<class T , class CT = const T>
T fflas_const_cast (
 CT x)

```

#### 11.21.3.206 failure()

```

Failure & failure () [inline]

```

#### 11.21.3.207 isOdd() [1/3]

```

template<class T >
bool isOdd (
 const T & a) [inline]

```

#### 11.21.3.208 isOdd() [2/3]

```

bool isOdd (
 const float & a) [inline]

```

#### 11.21.3.209 isOdd() [3/3]

```

bool isOdd (
 const double & a) [inline]

```

**11.21.3.210 NonZeroRandomMatrix()** [1/2]

```
template<class Field , class RandIter >
Field::Element_ptr NonZeroRandomMatrix (
 const Field & F,
 size_t m,
 size_t n,
 typename Field::Element_ptr A,
 size_t lda,
 RandIter & G) [inline]
```

Random non-zero Matrix.

Creates a  $m \times n$  matrix with random entries, and at least one of them is non zero.

**Parameters**

|     |       |                                                                     |
|-----|-------|---------------------------------------------------------------------|
|     | $F$   | field                                                               |
|     | $m$   | number of rows in $A$                                               |
|     | $n$   | number of cols in $A$                                               |
| out | $A$   | the matrix (preallocated to at least $m \times lda$ field elements) |
|     | $lda$ | leading dimension of $A$                                            |
|     | $G$   | a random iterator                                                   |

**Returns**

$A$ .

**11.21.3.211 NonZeroRandomMatrix()** [2/2]

```
template<class Field , class RandIter >
Field::Element_ptr NonZeroRandomMatrix (
 const Field & F,
 size_t m,
 size_t n,
 typename Field::Element_ptr A,
 size_t lda) [inline]
```

Random non-zero Matrix.

Creates a  $m \times n$  matrix with random entries, and at least one of them is non zero.

**Parameters**

|     |       |                                                                     |
|-----|-------|---------------------------------------------------------------------|
|     | $F$   | field                                                               |
|     | $m$   | number of rows in $A$                                               |
|     | $n$   | number of cols in $A$                                               |
| out | $A$   | the matrix (preallocated to at least $m \times lda$ field elements) |
|     | $lda$ | leading dimension of $A$                                            |

**Returns**

A.

**11.21.3.212 RandomMatrix() [1/2]**

```
template<class Field , class RandIter >
Field::Element_ptr RandomMatrix (
 const Field & F,
 size_t m,
 size_t n,
 typename Field::Element_ptr A,
 size_t lda,
 RandIter & G) [inline]
```

Random Matrix.

Creates a  $m \times n$  matrix with random entries.

**Parameters**

|     |            |                                                                     |
|-----|------------|---------------------------------------------------------------------|
|     | <i>F</i>   | field                                                               |
|     | <i>m</i>   | number of rows in A                                                 |
|     | <i>n</i>   | number of cols in A                                                 |
| out | <i>A</i>   | the matrix (preallocated to at least $m \times lda$ field elements) |
|     | <i>lda</i> | leading dimension of A                                              |
|     | <i>G</i>   | a random iterator                                                   |

**Returns**

A.

**11.21.3.213 RandomMatrix() [2/2]**

```
template<class Field >
Field::Element_ptr RandomMatrix (
 const Field & F,
 size_t m,
 size_t n,
 typename Field::Element_ptr A,
 size_t lda) [inline]
```

Random Matrix.

Creates a  $m \times n$  matrix with random entries.

**Parameters**

|     |            |                                                                     |
|-----|------------|---------------------------------------------------------------------|
|     | <i>F</i>   | field                                                               |
|     | <i>m</i>   | number of rows in A                                                 |
|     | <i>n</i>   | number of cols in A                                                 |
| out | <i>A</i>   | the matrix (preallocated to at least $m \times lda$ field elements) |
|     | <i>lda</i> | leading dimension of A                                              |

## Returns

A.

## 11.21.3.214 RandomTriangularMatrix() [1/2]

```
template<class Field , class RandIter >
Field::Element_ptr RandomTriangularMatrix (
 const Field & F,
 size_t m,
 size_t n,
 const FFLAS::FFLAS_UPLO UpLo,
 const FFLAS::FFLAS_DIAG Diag,
 bool nonsingular,
 typename Field::Element_ptr A,
 size_t lda,
 RandIter & G) [inline]
```

Random Triangular Matrix.

Creates a  $m \times n$  triangular matrix with random entries. The `UpLo` parameter defines whether it is upper or lower triangular.

## Parameters

|     |             |                                                                     |
|-----|-------------|---------------------------------------------------------------------|
|     | <i>F</i>    | field                                                               |
|     | <i>m</i>    | number of rows in A                                                 |
|     | <i>n</i>    | number of cols in A                                                 |
|     | <i>UpLo</i> | whether A is upper or lower triangular                              |
| out | <i>A</i>    | the matrix (preallocated to at least $m \times lda$ field elements) |
|     | <i>lda</i>  | leading dimension of A                                              |
|     | <i>G</i>    | a random iterator                                                   |

## Returns

A.

## 11.21.3.215 RandomTriangularMatrix() [2/2]

```
template<class Field >
Field::Element_ptr RandomTriangularMatrix (
 const Field & F,
 size_t m,
 size_t n,
 const FFLAS::FFLAS_UPLO UpLo,
 const FFLAS::FFLAS_DIAG Diag,
 bool nonsingular,
 typename Field::Element_ptr A,
 size_t lda) [inline]
```

Random Triangular Matrix.

Creates a  $m \times n$  triangular matrix with random entries. The `UpLo` parameter defines whether it is upper or lower triangular.

## Parameters

|     |        |                                                                     |
|-----|--------|---------------------------------------------------------------------|
|     | $F$    | field                                                               |
|     | $m$    | number of rows in $A$                                               |
|     | $n$    | number of cols in $A$                                               |
|     | $UpLo$ | whether $A$ is upper or lower triangular                            |
| out | $A$    | the matrix (preallocated to at least $m \times lda$ field elements) |
|     | $lda$  | leading dimension of $A$                                            |

## Returns

$A$ .

## 11.21.3.216 RandInt()

```
size_t RandInt (
 size_t a,
 size_t b) [inline]
```

## 11.21.3.217 RandomSymmetricMatrix()

```
template<class Field , class RandIter >
Field::Element_ptr RandomSymmetricMatrix (
 const Field & F,
 size_t n,
 bool nonsingular,
 typename Field::Element_ptr A,
 size_t lda,
 RandIter & G) [inline]
```

Random Symmetric Matrix.

Creates a  $m \times n$  triangular matrix with random entries. The  $UpLo$  parameter defines whether it is upper or lower triangular.

## Parameters

|     |       |                                                                     |
|-----|-------|---------------------------------------------------------------------|
|     | $F$   | field                                                               |
|     | $n$   | order of $A$                                                        |
| out | $A$   | the matrix (preallocated to at least $n \times lda$ field elements) |
|     | $lda$ | leading dimension of $A$                                            |
|     | $G$   | a random iterator                                                   |

## Returns

$A$ .



**11.21.3.218 RandomMatrixWithRank()** [1/2]

```
template<class Field , class RandIter >
Field::Element_ptr RandomMatrixWithRank (
 const Field & F,
 size_t m,
 size_t n,
 size_t r,
 typename Field::Element_ptr A,
 size_t lda,
 RandIter & G) [inline]
```

Random Matrix with prescribed rank.

Creates an  $m \times n$  matrix with random entries and rank  $r$ .

**Parameters**

|       |                                                                     |
|-------|---------------------------------------------------------------------|
| $F$   | field                                                               |
| $m$   | number of rows in $A$                                               |
| $n$   | number of cols in $A$                                               |
| $r$   | rank of the matrix to build                                         |
| $A$   | the matrix (preallocated to at least $m \times lda$ field elements) |
| $lda$ | leading dimension of $A$                                            |
| $G$   | a random iterator                                                   |

**Returns**

$A$ .

**11.21.3.219 RandomMatrixWithRank()** [2/2]

```
template<class Field >
Field::Element_ptr RandomMatrixWithRank (
 const Field & F,
 size_t m,
 size_t n,
 size_t r,
 typename Field::Element_ptr A,
 size_t lda) [inline]
```

Random Matrix with prescribed rank.

Creates an  $m \times n$  matrix with random entries and rank  $r$ .

**Parameters**

|     |       |                                                                     |
|-----|-------|---------------------------------------------------------------------|
|     | $F$   | field                                                               |
|     | $m$   | number of rows in $A$                                               |
|     | $n$   | number of cols in $A$                                               |
|     | $r$   | rank of the matrix to build                                         |
| out | $A$   | the matrix (preallocated to at least $m \times lda$ field elements) |
|     | $lda$ | leading dimension of $A$                                            |

## Returns

A.

**11.21.3.220 RandomIndexSubset()**

```
size_t * RandomIndexSubset (
 size_t N,
 size_t R,
 size_t * P) [inline]
```

Pick uniformly at random a sequence of  $R$  distinct elements from the set  $\{0, \dots, N - 1\}$  using Knuth's shuffle.

## Parameters

|     |     |                                                             |
|-----|-----|-------------------------------------------------------------|
|     | $N$ | the cardinality of the sampling set                         |
|     | $R$ | the number of elements to sample                            |
| out | $P$ | the output sequence (pre-allocated to at least $R$ indices) |

**11.21.3.221 RandomPermutation()**

```
size_t * RandomPermutation (
 size_t N,
 size_t * P) [inline]
```

Pick uniformly at random a permutation of size  $N$  stored in LAPACK format using Knuth's shuffle.

## Parameters

|     |     |                                                                |
|-----|-----|----------------------------------------------------------------|
|     | $N$ | the length of the permutation                                  |
| out | $P$ | the output permutation (pre-allocated to at least $N$ indices) |

**11.21.3.222 RandomRankProfileMatrix()**

```
void RandomRankProfileMatrix (
 size_t M,
 size_t N,
 size_t R,
 size_t * rows,
 size_t * cols) [inline]
```

Pick uniformly at random an  $R$ -subpermutation of dimension  $M \times N$  : a matrix with only  $R$  non-zeros equal to one, in a random rook placement.

## Parameters

|     |        |                                                              |
|-----|--------|--------------------------------------------------------------|
|     | $M$    | row dimension                                                |
|     | $N$    | column dimension                                             |
| out | $rows$ | the row position of each non zero element (pre-allocated)    |
| out | $cols$ | the column position of each non zero element (pre-allocated) |

**11.21.3.223 swapval()**

```
void swapval (
 size_t k,
 size_t N,
 size_t * P,
 size_t val) [inline]
```

**11.21.3.224 RandomSymmetricRankProfileMatrix()**

```
void RandomSymmetricRankProfileMatrix (
 size_t N,
 size_t R,
 size_t * rows,
 size_t * cols) [inline]
```

Pick uniformly at random a symmetric R-subpermutation of dimension  $N \times N$  : a symmetric matrix with only R non-zeros, all equal to one, in a random rook placement.

**Parameters**

|     |             |                                                              |
|-----|-------------|--------------------------------------------------------------|
|     | <i>N</i>    | matrix order                                                 |
| out | <i>rows</i> | the row position of each non zero element (pre-allocated)    |
| out | <i>cols</i> | the column position of each non zero element (pre-allocated) |

**11.21.3.225 RandomLTQSRankProfileMatrix()**

```
void RandomLTQSRankProfileMatrix (
 size_t n,
 size_t r,
 size_t t,
 size_t * rows,
 size_t * cols) [inline]
```

**11.21.3.226 RandomMatrixWithRankandRPM()** [1/2]

```
template<class Field , class RandIter >
Field::Element_ptr RandomMatrixWithRankandRPM (
 const Field & F,
 size_t M,
 size_t N,
 size_t R,
 typename Field::Element_ptr A,
 size_t lda,
 const size_t * RRP,
 const size_t * CRP,
 RandIter & G) [inline]
```

Random Matrix with prescribed rank and rank profile matrix Creates an  $m \times n$  matrix with random entries and rank  $r$ .

## Parameters

|            |                                                                                  |
|------------|----------------------------------------------------------------------------------|
| <i>F</i>   | field                                                                            |
| <i>m</i>   | number of rows in A                                                              |
| <i>n</i>   | number of cols in A                                                              |
| <i>r</i>   | rank of the matrix to build                                                      |
| <i>A</i>   | the matrix (preallocated to at least $m \times lda$ field elements)              |
| <i>lda</i> | leading dimension of A                                                           |
| <i>RRP</i> | the R dimensional array with row positions of the rank profile matrix' pivots    |
| <i>CRP</i> | the R dimensional array with column positions of the rank profile matrix' pivots |
| <i>G</i>   | a random iterator                                                                |

## Returns

A.

## 11.21.3.227 RandomMatrixWithRankandRPM() [2/2]

```
template<class Field >
Field::Element_ptr RandomMatrixWithRankandRPM (
 const Field & F,
 size_t M,
 size_t N,
 size_t R,
 typename Field::Element_ptr A,
 size_t lda,
 const size_t * RRP,
 const size_t * CRP) [inline]
```

Random Matrix with prescribed rank and rank profile matrix Creates an  $m \times n$  matrix with random entries and rank  $r$ .

## Parameters

|            |                                                                                  |
|------------|----------------------------------------------------------------------------------|
| <i>F</i>   | field                                                                            |
| <i>m</i>   | number of rows in A                                                              |
| <i>n</i>   | number of cols in A                                                              |
| <i>r</i>   | rank of the matrix to build                                                      |
| <i>A</i>   | the matrix (preallocated to at least $m \times lda$ field elements)              |
| <i>lda</i> | leading dimension of A                                                           |
| <i>RRP</i> | the R dimensional array with row positions of the rank profile matrix' pivots    |
| <i>CRP</i> | the R dimensional array with column positions of the rank profile matrix' pivots |

## Returns

A.

## 11.21.3.228 RandomSymmetricMatrixWithRankandRPM() [1/2]

```
template<class Field , class RandIter >
Field::Element_ptr RandomSymmetricMatrixWithRankandRPM (
```

```

const Field & F,
size_t N,
size_t R,
typename Field::Element_ptr A,
size_t lda,
const size_t * RRP,
const size_t * CRP,
RandIter & G) [inline]

```

Random Symmetric Matrix with prescribed rank and rank profile matrix Creates an  $n \times n$  symmetric matrix with random entries and rank  $r$ .

#### Parameters

|            |                                                                                  |
|------------|----------------------------------------------------------------------------------|
| <i>F</i>   | field                                                                            |
| <i>n</i>   | order of A                                                                       |
| <i>r</i>   | rank of A                                                                        |
| <i>A</i>   | the matrix (preallocated to at least $n \times lda$ field elements)              |
| <i>lda</i> | leading dimension of A                                                           |
| <i>RRP</i> | the R dimensional array with row positions of the rank profile matrix' pivots    |
| <i>CRP</i> | the R dimensional array with column positions of the rank profile matrix' pivots |
| <i>G</i>   | a random iterator                                                                |

#### Returns

A.

### 11.21.3.229 RandomSymmetricMatrixWithRankandRPM() [2/2]

```

template<class Field >
Field::Element_ptr RandomSymmetricMatrixWithRankandRPM (
 const Field & F,
 size_t M,
 size_t N,
 size_t R,
 typename Field::Element_ptr A,
 size_t lda,
 const size_t * RRP,
 const size_t * CRP) [inline]

```

Random Symmetric Matrix with prescribed rank and rank profile matrix Creates an  $n \times n$  symmetric matrix with random entries and rank  $r$ .

#### Parameters

|            |                                                                                  |
|------------|----------------------------------------------------------------------------------|
| <i>F</i>   | field                                                                            |
| <i>n</i>   | order of A                                                                       |
| <i>r</i>   | rank of A                                                                        |
| <i>A</i>   | the matrix (preallocated to at least $n \times lda$ field elements)              |
| <i>lda</i> | leading dimension of A                                                           |
| <i>RRP</i> | the R dimensional array with row positions of the rank profile matrix' pivots    |
| <i>CRP</i> | the R dimensional array with column positions of the rank profile matrix' pivots |

## Returns

$A$ .

### 11.21.3.230 RandomMatrixWithRankandRandomRPM() [1/2]

```
template<class Field , class RandIter >
Field::Element_ptr RandomMatrixWithRankandRandomRPM (
 const Field & F,
 size_t M,
 size_t N,
 size_t R,
 typename Field::Element_ptr A,
 size_t lda,
 RandIter & G) [inline]
```

Random Matrix with prescribed rank, with random rank profile matrix Creates an  $m \times n$  matrix with random entries, rank  $r$  and with a rank profile matrix chosen uniformly at random.

## Parameters

|       |                                                                     |
|-------|---------------------------------------------------------------------|
| $F$   | field                                                               |
| $m$   | number of rows in $A$                                               |
| $n$   | number of cols in $A$                                               |
| $r$   | rank of the matrix to build                                         |
| $A$   | the matrix (preallocated to at least $m \times lda$ field elements) |
| $lda$ | leading dimension of $A$                                            |
| $G$   | a random iterator                                                   |

## Returns

$A$ .

### 11.21.3.231 RandomMatrixWithRankandRandomRPM() [2/2]

```
template<class Field >
Field::Element_ptr RandomMatrixWithRankandRandomRPM (
 const Field & F,
 size_t M,
 size_t N,
 size_t R,
 typename Field::Element_ptr A,
 size_t lda) [inline]
```

Random Matrix with prescribed rank, with random rank profile matrix Creates an  $m \times n$  matrix with random entries, rank  $r$  and with a rank profile matrix chosen uniformly at random.

## Parameters

|       |                                                                     |
|-------|---------------------------------------------------------------------|
| $F$   | field                                                               |
| $m$   | number of rows in $A$                                               |
| $n$   | number of cols in $A$                                               |
| $r$   | rank of the matrix to build                                         |
| $A$   | the matrix (preallocated to at least $m \times lda$ field elements) |
| $lda$ | leading dimension of $A$                                            |

## Returns

A.

**11.21.3.232 RandomSymmetricMatrixWithRankandRandomRPM()** [1/2]

```
template<class Field , class RandIter >
Field::Element_ptr RandomSymmetricMatrixWithRankandRandomRPM (
 const Field & F,
 size_t N,
 size_t R,
 typename Field::Element_ptr A,
 size_t lda,
 RandIter & G) [inline]
```

Random Symmetric Matrix with prescribed rank, with random rank profile matrix Creates an  $n \times n$  matrix with random entries, rank  $r$  and with a rank profile matrix chosen uniformly at random.

## Parameters

|       |                                                                     |
|-------|---------------------------------------------------------------------|
| $F$   | field                                                               |
| $n$   | order of A                                                          |
| $r$   | rank of A                                                           |
| $A$   | the matrix (preallocated to at least $n \times lda$ field elements) |
| $lda$ | leading dimension of A                                              |
| $G$   | a random iterator                                                   |

## Returns

A.

**11.21.3.233 RandomSymmetricMatrixWithRankandRandomRPM()** [2/2]

```
template<class Field >
Field::Element_ptr RandomSymmetricMatrixWithRankandRandomRPM (
 const Field & F,
 size_t N,
 size_t R,
 typename Field::Element_ptr A,
 size_t lda) [inline]
```

Random Symmetric Matrix with prescribed rank, with random rank profile matrix Creates an  $n \times n$  matrix with random entries, rank  $r$  and with a rank profile matrix chosen uniformly at random.

## Parameters

|       |                                                                     |
|-------|---------------------------------------------------------------------|
| $F$   | field                                                               |
| $n$   | order of A                                                          |
| $r$   | rank of A                                                           |
| $A$   | the matrix (preallocated to at least $n \times lda$ field elements) |
| $lda$ | leading dimension of A                                              |

**Returns**

A.

**11.21.3.234 RandomMatrixWithDet() [1/2]**

```
template<class Field >
Field::Element_ptr RandomMatrixWithDet (
 const Field & F,
 size_t n,
 const typename Field::Element d,
 typename Field::Element_ptr A,
 size_t lda) [inline]
```

Random Matrix with prescribed det.

Creates a  $m \times n$  matrix with random entries and rank  $r$ .

**Parameters**

|       |                                                                                     |
|-------|-------------------------------------------------------------------------------------|
| $F$   | field                                                                               |
| $d$   | the prescribed value for the determinant of A                                       |
| $n$   | number of cols in A                                                                 |
| $A$   | the matrix to be generated (preallocated to at least $n \times lda$ field elements) |
| $lda$ | leading dimension of A                                                              |

**Returns**

A.

**11.21.3.235 RandomMatrixWithDet() [2/2]**

```
template<class Field , class RandIter >
Field::Element_ptr RandomMatrixWithDet (
 const Field & F,
 size_t n,
 const typename Field::Element d,
 typename Field::Element_ptr A,
 size_t lda,
 RandIter & G) [inline]
```

Random Matrix with prescribed det.

Creates a  $m \times n$  matrix with random entries and rank  $r$ .

**Parameters**

|       |                                                                                     |
|-------|-------------------------------------------------------------------------------------|
| $F$   | field                                                                               |
| $d$   | the prescribed value for the determinant of A                                       |
| $n$   | number of cols in A                                                                 |
| $A$   | the matrix to be generated (preallocated to at least $n \times lda$ field elements) |
| $lda$ | leading dimension of A                                                              |



## Returns

A.

**11.21.3.236 RandomLTQSMATRIXWithRankandQSOrder()**

```
template<class Field , class RandIter >
Field::Element_ptr RandomLTQSMATRIXWithRankandQSOrder (
 Field & F,
 size_t n,
 size_t r,
 size_t t,
 typename Field::Element_ptr A,
 size_t lda,
 RandIter & G) [inline]
```

**11.21.3.237 chooseField()**

```
template<typename Field >
Field * chooseField (
 Givaro::Integer q,
 uint64_t b,
 uint64_t seed)
```

**11.21.3.238 chooseField< Givaro::ZRing< int32\_t > >()**

```
template<>
Givaro::ZRing< int32_t > * chooseField< Givaro::ZRing< int32_t > > (
 Givaro::Integer q,
 uint64_t b,
 uint64_t seed)
```

**11.21.3.239 chooseField< Givaro::ZRing< int64\_t > >()**

```
template<>
Givaro::ZRing< int64_t > * chooseField< Givaro::ZRing< int64_t > > (
 Givaro::Integer q,
 uint64_t b,
 uint64_t seed)
```

**11.21.3.240 chooseField< Givaro::ZRing< float > >()**

```
template<>
Givaro::ZRing< float > * chooseField< Givaro::ZRing< float > > (
 Givaro::Integer q,
 uint64_t b,
 uint64_t seed)
```

### 11.21.3.241 chooseField< Givaro::ZRing< double > >()

```
template<>
Givaro::ZRing< double > * chooseField< Givaro::ZRing< double > > (
 Givaro::Integer q,
 uint64_t b,
 uint64_t seed)
```

## 11.22 FFPACK::Protected Namespace Reference

### Functions

- [template<class Field >](#)  
[size\\_t LUdivine\\_construct](#) (const Field &F, const FFLAS::FFLAS\_DIAG Diag, const size\_t M, const size\_t N, typename Field::ConstElement\_ptr A, const size\_t lda, typename Field::Element\_ptr X, const size\_t ldx, typename Field::Element\_ptr u, const size\_t incu, size\_t \*P, bool computeX, const FFPACK\_MINPOLY\_TAG MinTag=FFpackDense, const size\_t kg\_mc=0, const size\_t kg\_mb=0, const size\_t kg\_j=0)
- [template<class Field >](#)  
[size\\_t GaussJordan](#) (const Field &F, const size\_t M, const size\_t N, typename Field::Element\_ptr A, const size\_t lda, const size\_t colbeg, const size\_t rowbeg, const size\_t colsize, size\_t \*P, size\_t \*Q, const FFPACK::FFPACK\_LU\_TAG LuTag)  
*Gauss-Jordan algorithm computing the Reduced Row echelon form and its transform matrix.*
- [template<class Field , class Polynomial >](#)  
[std::list< Polynomial > & KellerGehrig](#) (const Field &F, std::list< Polynomial > &charp, const size\_t N, typename Field::ConstElement\_ptr A, const size\_t lda)
- [template<class Field , class Polynomial >](#)  
[int KGFast](#) (const Field &F, std::list< Polynomial > &charp, const size\_t N, typename Field::Element\_ptr A, const size\_t lda, size\_t \*kg\_mc, size\_t \*kg\_mb, size\_t \*kg\_j)
- [template<class Field , class Polynomial >](#)  
[std::list< Polynomial > & KGFast\\_generalized](#) (const Field &F, std::list< Polynomial > &charp, const size\_t N, typename Field::Element\_ptr A, const size\_t lda)
- [template<class Field >](#)  
[void fgemv\\_kgf](#) (const Field &F, const size\_t N, typename Field::ConstElement\_ptr A, const size\_t lda, typename Field::ConstElement\_ptr X, const size\_t incX, typename Field::Element\_ptr Y, const size\_t incY, const size\_t kg\_mc, const size\_t kg\_mb, const size\_t kg\_j)
- [template<class Field , class Polynomial , class RandIter >](#)  
[std::list< Polynomial > & LUKrylov](#) (const Field &F, std::list< Polynomial > &charp, const size\_t N, typename Field::Element\_ptr A, const size\_t lda, typename Field::Element\_ptr U, const size\_t ldu, RandIter &G)
- [template<class Field , class Polynomial >](#)  
[std::list< Polynomial > & Danilevski](#) (const Field &F, std::list< Polynomial > &charp, const size\_t N, typename Field::Element\_ptr A, const size\_t lda)
- [template<class PolRing >](#)  
[void RandomKrylovPrecond](#) (const PolRing &PR, std::list< typename PolRing::Element > &completedFactors, const size\_t N, typename PolRing::Domain\_t::Element\_ptr A, const size\_t lda, size\_t &Nb, typename PolRing::Domain\_t::Element\_ptr &B, size\_t &ldb, typename PolRing::Domain\_t::RandIter &g, const size\_t degree=\_\_FFLASFFPACK\_ARITHPROG\_THRESHOLD)
- [template<class PolRing >](#)  
[std::list< typename PolRing::Element > & ArithProg](#) (const PolRing &PR, std::list< typename PolRing::Element > &frobeniusForm, const size\_t N, typename PolRing::Domain\_t::Element\_ptr A, const size\_t lda, const size\_t degree)
- [template<class Field , class Polynomial >](#)  
[std::list< Polynomial > & LUKrylov\\_KGFast](#) (const Field &F, std::list< Polynomial > &charp, const size\_t N, typename Field::Element\_ptr A, const size\_t lda, typename Field::Element\_ptr X, const size\_t ldx)

- `template<class Field , class Polynomial >`  
`Polynomial & MatVecMinPoly (const Field &F, Polynomial &minP, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr v, const size_t incv, typename Field::Element_ptr K, const size_t ldk, size_t *P)`
- `template<class Field , class Polynomial >`  
`Polynomial & Hybrid_KGF_LUK_MinPoly (const Field &F, Polynomial &minP, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr X, const size_t ldx, size_t *P, const FFPACK_MINPOLY_TAG MinTag=FFPACK::FfpackDense, const size_t kg_mc=0, const size_t kg_mb=0, const size_t kg_j=0)`
- `template<class Field >`  
`size_t updatedD (const Field &F, size_t *d, size_t k, std::vector< std::vector< typename Field::Element > > &minpt)`
- `template<class Field >`  
`size_t newD (const Field &F, size_t *d, bool &KeepOn, const size_t l, const size_t N, typename Field::Element_ptr X, const size_t *Q, std::vector< std::vector< typename Field::Element > > &minpt)`
- `template<class Field >`  
`void CompressRows (Field &F, const size_t M, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr tmp, const size_t ldtmp, const size_t *d, const size_t nb_blocs)`
- `template<class Field >`  
`void CompressRowsQK (Field &F, const size_t M, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr tmp, const size_t ldtmp, const size_t *d, const size_t deg, const size_t nb_blocs)`
- `template<class Field >`  
`void DeCompressRows (Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr tmp, const size_t ldtmp, const size_t *d, const size_t nb_blocs)`
- `template<class Field >`  
`void DeCompressRowsQK (Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr tmp, const size_t ldtmp, const size_t *d, const size_t deg, const size_t nb_blocs)`
- `template<class Field >`  
`void CompressRowsQA (Field &F, const size_t M, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr tmp, const size_t ldtmp, const size_t *d, const size_t nb_blocs)`
- `template<class Field >`  
`void DeCompressRowsQA (Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr tmp, const size_t ldtmp, const size_t *d, const size_t nb_blocs)`
- `template<class Field >`  
`size_t LUdivine_construct (const Field &F, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr X, const size_t ldx, typename Field::Element_ptr u, const size_t incu, size_t *P, bool computeX, const FFPACK::FFPACK_MINPOLY_TAG MinTag, const size_t kg_mc, const size_t kg_mb, const size_t kg_j)`

## 11.22.1 Function Documentation

### 11.22.1.1 LUdivine\_construct() [1/2]

```
template<class Field >
size_t LUdivine_construct (
 const Field & F,
 const FFLAS::FFLAS_DIAG Diag,
 const size_t M,
 const size_t N,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::Element_ptr X,
 const size_t ldx,
 typename Field::Element_ptr u,
```

```

const size_t incu,
size_t * P,
bool computeX,
const FFPACK_MINPOLY_TAG MinTag = FfpackDense,
const size_t kq_mc = 0,
const size_t kq_mb = 0,
const size_t kq_j = 0)

```

### 11.22.1.2 GaussJordan()

```

template<class Field >
size_t GaussJordan (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 const size_t colbeg,
 const size_t rowbeg,
 const size_t colsize,
 size_t * P,
 size_t * Q,
 const FFPACK::FFPACK_LU_TAG LuTag) [inline]

```

Gauss-Jordan algorithm computing the Reduced Row echelon form and its transform matrix.

- Bibliography**
- Algorithm 2.8 of A. Storjohann Thesis 2000,
  - Algorithm 11 of Jeannerod C-P., Pernet, C. and Storjohann, A. *Rank-profile revealing Gaussian elimination and the CUP matrix decomposition*, J. of Symbolic Comp., 2013

#### Parameters

|                |              |                                                                                                          |
|----------------|--------------|----------------------------------------------------------------------------------------------------------|
|                | <i>M</i>     | row dimension of A                                                                                       |
|                | <i>N</i>     | column dimension of A                                                                                    |
| <i>in, out</i> | <i>A</i>     | an m x n matrix                                                                                          |
|                | <i>lda</i>   | leading dimension of A                                                                                   |
|                | <i>P</i>     | row permutation                                                                                          |
|                | <i>Q</i>     | column permutation                                                                                       |
|                | <i>LuTag</i> | set the base case to a Tile (FfpackGaussJordanTile) or Slab (FfpackGaussJordanSlab) recursive RedEchelon |

| I | A11 | A12 | | | | | | | | I | \* | A22 | | | 0 | 0 | A22 | | | | | | | | 0 | A32 | | | | | | | |  
where the transformation matrix is stored at the pivot column position

### 11.22.1.3 KellerGehrig()

```

template<class Field , class Polynomial >
std::list< Polynomial > & KellerGehrig (
 const Field & F,
 std::list< Polynomial > & charp,
 const size_t N,
 typename Field::ConstElement_ptr A,
 const size_t lda)

```

**11.22.1.4 KGFast()**

```
template<class Field , class Polynomial >
int KGFast (
 const Field & F,
 std::list< Polynomial > & charp,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t * kg_mc,
 size_t * kg_mb,
 size_t * kg_j)
```

**11.22.1.5 KGFast\_generalized()**

```
template<class Field , class Polynomial >
std::list< Polynomial > & KGFast_generalized (
 const Field & F,
 std::list< Polynomial > & charp,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda)
```

**11.22.1.6 fgemv\_kgf()**

```
template<class Field >
void fgemv_kgf (
 const Field & F,
 const size_t N,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr X,
 const size_t incX,
 typename Field::Element_ptr Y,
 const size_t incY,
 const size_t kg_mc,
 const size_t kg_mb,
 const size_t kg_j)
```

**11.22.1.7 LUKrylov()**

```
template<class Field , class Polynomial , class RandIter >
std::list< Polynomial > & LUKrylov (
 const Field & F,
 std::list< Polynomial > & charp,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr U,
 const size_t ldu,
 RandIter & G)
```

**11.22.1.8 Danilevski()**

```
template<class Field , class Polynomial >
std::list< Polynomial > & Danilevski (
 const Field & F,
 std::list< Polynomial > & charp,
 const size_t N,
```

```

 typename Field::Element_ptr A,
 const size_t lda)

```

#### 11.22.1.9 RandomKrylovPrecond()

```

template<class PolRing >
void RandomKrylovPrecond (
 const PolRing & PR,
 std::list< typename PolRing::Element > & completedFactors,
 const size_t N,
 typename PolRing::Domain_t::Element_ptr A,
 const size_t lda,
 size_t & Nb,
 typename PolRing::Domain_t::Element_ptr & B,
 size_t & ldb,
 typename PolRing::Domain_t::RandIter & g,
 const size_t degree = __FFLASFFPACK_ARITHPROG_THRESHOLD) [inline]

```

**Todo** swap to save space ??

**Todo**

**Todo** don't assing K2 c\*noc x N but only mas (c,noc) x N and store each one after the other

**Todo** swap to save space ??

**Todo**

**Todo** don't assing K2 c\*noc x N but only mas (c,noc) x N and store each one after the other

#### 11.22.1.10 ArithProg()

```

template<class PolRing >
std::list< typename PolRing::Element > & ArithProg (
 const PolRing & PR,
 std::list< typename PolRing::Element > & frobeniusForm,
 const size_t N,
 typename PolRing::Domain_t::Element_ptr A,
 const size_t lda,
 const size_t degree) [inline]

```

#### 11.22.1.11 LUKrylov\_KGFast()

```

template<class Field , class Polynomial >
std::list< Polynomial > & LUKrylov_KGFast (
 const Field & F,
 std::list< Polynomial > & charp,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr X,
 const size_t ldx)

```

#### 11.22.1.12 MatVecMinPoly()

```

template<class Field , class Polynomial >
Polynomial & MatVecMinPoly (
 const Field & F,

```

```

 Polynomial & minP,
 const size_t N,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::Element_ptr v,
 const size_t incv,
 typename Field::Element_ptr K,
 const size_t ldk,
 size_t * P) [inline]

```

#### 11.22.1.13 Hybrid\_KGF\_LUK\_MinPoly()

```

template<class Field , class Polynomial >
Polynomial & Hybrid_KGF_LUK_MinPoly (
 const Field & F,
 Polynomial & minP,
 const size_t N,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::Element_ptr X,
 const size_t ldx,
 size_t * P,
 const FFPACK_MINPOLY_TAG MinTag = FFPACK::FfpackDense,
 const size_t kg_mc = 0,
 const size_t kg_mb = 0,
 const size_t kg_j = 0)

```

#### 11.22.1.14 updateD()

```

template<class Field >
size_t updateD (
 const Field & F,
 size_t * d,
 size_t k,
 std::vector< std::vector< typename Field::Element > > & minpt)

```

#### 11.22.1.15 newD()

```

template<class Field >
size_t newD (
 const Field & F,
 size_t * d,
 bool & KeepOn,
 const size_t l,
 const size_t N,
 typename Field::Element_ptr X,
 const size_t * Q,
 std::vector< std::vector< typename Field::Element > > & minpt)

```

#### 11.22.1.16 CompressRows()

```

template<class Field >
void CompressRows (
 Field & F,
 const size_t M,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr tmp,
 const size_t ldtmp,

```

```

 const size_t * d,
 const size_t nb_blocs) [inline]

```

#### 11.22.1.17 CompressRowsQK()

```

template<class Field >
void CompressRowsQK (
 Field & F,
 const size_t M,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr tmp,
 const size_t ldtmp,
 const size_t * d,
 const size_t deg,
 const size_t nb_blocs) [inline]

```

#### 11.22.1.18 DeCompressRows()

```

template<class Field >
void DeCompressRows (
 Field & F,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr tmp,
 const size_t ldtmp,
 const size_t * d,
 const size_t nb_blocs) [inline]

```

#### 11.22.1.19 DeCompressRowsQK()

```

template<class Field >
void DeCompressRowsQK (
 Field & F,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr tmp,
 const size_t ldtmp,
 const size_t * d,
 const size_t deg,
 const size_t nb_blocs) [inline]

```

#### 11.22.1.20 CompressRowsQA()

```

template<class Field >
void CompressRowsQA (
 Field & F,
 const size_t M,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr tmp,
 const size_t ldtmp,
 const size_t * d,
 const size_t nb_blocs) [inline]

```



**11.22.1.21 DeCompressRowsQA()**

```
template<class Field >
void DeCompressRowsQA (
 Field & F,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 typename Field::Element_ptr tmp,
 const size_t ldtmp,
 const size_t * d,
 const size_t nb_blocs) [inline]
```

**11.22.1.22 LUdivine\_construct() [2/2]**

```
template<class Field >
size_t LUdivine_construct (
 const Field & F,
 const FFLAS::FFLAS_DIAG Diag,
 const size_t M,
 const size_t N,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::Element_ptr X,
 const size_t ldx,
 typename Field::Element_ptr u,
 const size_t incu,
 size_t * P,
 bool computeX,
 const FFPACK::FFPACK_MINPOLY_TAG MinTag,
 const size_t kg_mc,
 const size_t kg_mb,
 const size_t kg_j)
```

**11.23 Givaro Namespace Reference****Data Structures**

- class [ModularBalanced](#)
- class [Montgomery](#)

**11.24 MKL\_CONFIG Namespace Reference****11.25 Reclnt Namespace Reference****Data Structures**

- class [rint](#)
- class [ruint](#)



# Chapter 12

## Data Structure Documentation

### 12.1 AlgoChooser< ModeT, ParSeq > Struct Template Reference

#### Public Types

- [typedef MMHelperAlgo::Winograd value](#)

#### 12.1.1 Member Typedef Documentation

##### 12.1.1.1 value

```
template<class ModeT , class ParSeq >
typedef MMHelperAlgo::Winograd value
```

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

### 12.2 AlgoChooser< ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeq > Struct Template Reference

#### Public Types

- [typedef MMHelperAlgo::Classic value](#)

#### 12.2.1 Member Typedef Documentation

##### 12.2.1.1 value

```
template<class ParSeq >
typedef MMHelperAlgo::Classic value
```

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

### 12.3 ALL< v > Struct Template Reference

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.4 ALL< false, v... > Struct Template Reference

### Static Public Attributes

- [static constexpr bool value = false](#)

### 12.4.1 Field Documentation

#### 12.4.1.1 value

```
template<bool... v>
constexpr bool value = false [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.5 ALL< true, v... > Struct Template Reference

### Static Public Attributes

- [static constexpr bool value = ALL<v...>::value](#)

### 12.5.1 Field Documentation

#### 12.5.1.1 value

```
template<bool... v>
constexpr bool value = ALL<v...>::value [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.6 ALL<> Struct Reference

### Static Public Attributes

- [static constexpr bool value = true](#)

### 12.6.1 Field Documentation

#### 12.6.1.1 value

```
constexpr bool value = true [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.7 ArbitraryPrecIntTag Struct Reference

Arbitrary precision integers: GMP.

```
#include <field-traits.h>
```

### 12.7.1 Detailed Description

Arbitrary precision integers: GMP.

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.8 AreEqual< X, Y > Class Template Reference

```
#include <fflas_enum.h>
```

### Static Public Attributes

- `static const bool value = false`

### 12.8.1 Field Documentation

#### 12.8.1.1 value

```
template<class X , class Y >
const bool value = false [static]
```

The documentation for this class was generated from the following file:

- `fflas_enum.h`

## 12.9 AreEqual< X, X > Class Template Reference

```
#include <fflas_enum.h>
```

### Static Public Attributes

- `static const bool value = true`

### 12.9.1 Field Documentation

#### 12.9.1.1 value

```
template<class X >
const bool value = true [static]
```

The documentation for this class was generated from the following file:

- `fflas_enum.h`

## 12.10 Argument Struct Reference

```
#include <args-parser.h>
```

### Data Fields

- `char c`
- `const char * example`
- `const char * helpString`
- `ArgumentType type`
- `void * data`

### 12.10.1 Field Documentation

#### 12.10.1.1 c

```
char c
```

#### 12.10.1.2 example

```
const char* example
```

### 12.10.1.3 helpString

```
const char* helpString
```

### 12.10.1.4 type

```
ArgumentType type
```

### 12.10.1.5 data

```
void* data
```

The documentation for this struct was generated from the following file:

- [args-parser.h](#)

## 12.11 associatedDelayedField< Field > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef Field field](#)
- [typedef Field & type](#)

### 12.11.1 Member Typedef Documentation

#### 12.11.1.1 field

```
template<class Field >
typedef Field field
```

#### 12.11.1.2 type

```
template<class Field >
typedef Field& type
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.12 associatedDelayedField< const FFPACK::RNSIntegerMod< RNS > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef FFPACK::RNSInteger< RNS > field](#)
- [typedef FFPACK::RNSInteger< RNS > type](#)

### 12.12.1 Member Typedef Documentation

#### 12.12.1.1 field

```
template<typename RNS >
typedef FFPACK::RNSInteger<RNS> field
```

### 12.12.1.2 type

```
template<typename RNS >
typedef FFPACK::RNSInteger<RNS> type
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.13 associatedDelayedField< const Givaro::Modular< T, X > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef](#) Givaro::ZRing< T > [field](#)
- [typedef](#) Givaro::ZRing< T > [type](#)

### 12.13.1 Member Typedef Documentation

#### 12.13.1.1 field

```
template<typename T , typename X >
typedef Givaro::ZRing<T> field
```

#### 12.13.1.2 type

```
template<typename T , typename X >
typedef Givaro::ZRing<T> type
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.14 associatedDelayedField< const Givaro::ModularBalanced< T > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef](#) Givaro::ZRing< T > [field](#)
- [typedef](#) Givaro::ZRing< T > [type](#)

### 12.14.1 Member Typedef Documentation

#### 12.14.1.1 field

```
template<typename T >
typedef Givaro::ZRing<T> field
```

#### 12.14.1.2 type

```
template<typename T >
typedef Givaro::ZRing<T> type
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.15 associatedDelayedField< const Givaro::ZRing< T > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- `typedef` Givaro::ZRing< T > [field](#)
- `typedef` Givaro::ZRing< T > [type](#)

### 12.15.1 Member Typedef Documentation

#### 12.15.1.1 field

```
template<typename T >
typedef Givaro::ZRing<T> field
```

#### 12.15.1.2 type

```
template<typename T >
typedef Givaro::ZRing<T> type
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.16 Auto Struct Reference

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

## 12.17 Bench< Elt > Class Template Reference

### Public Types

- `using` [Field](#) = Modular< [Elt](#) >
- `using` [Elt\\_ptr](#) = `typename` [Field](#)::[Element\\_ptr](#)
- `using` [Residu](#) = `typename` [Field](#)::[Residu\\_t](#)
- `template<bool B, class T = void>`  
`using` [enable\\_if\\_t](#) = `typename` `std::enable_if< B, T >::type`
- `template<typename Simd >`  
`using` [is\\_same\\_element](#) = `typename` [Simd](#)::`template` [is\\_same\\_element](#)< [Field](#) >
- `template<typename E >`  
`using` [enable\\_if\\_no\\_simd\\_t](#) = [enable\\_if\\_t](#)< [Simd](#)< E >::[vect\\_size](#)==1 >
- `template<typename E >`  
`using` [enable\\_if\\_simd128\\_t](#) = [enable\\_if\\_t](#)< `sizeof`(E) \*[Simd](#)< E >::[vect\\_size](#)==16 >
- `template<typename E >`  
`using` [enable\\_if\\_simd256\\_t](#) = [enable\\_if\\_t](#)< `sizeof`(E) \*[Simd](#)< E >::[vect\\_size](#)==32 >
- `template<typename E >`  
`using` [enable\\_if\\_simd512\\_t](#) = [enable\\_if\\_t](#)< `sizeof`(E) \*[Simd](#)< E >::[vect\\_size](#)==64 >

### Public Member Functions

- [Bench](#) ([size\\_t](#) m, [size\\_t](#) n, [size\\_t](#) iters, `bool` inplace)
- `template<typename Simd = NoSimd<Elt>, enable_if_t< is_same_element< Simd >::value > * = nullptr>`  
`void` [doBenchs](#) ()
- `template<typename _E = Elt, enable_if_t< is_same< _E, Elt >::value > * = nullptr, enable_if_no_simd_t< _E > * = nullptr>`  
`void` [run](#) (`bool` allsimd)



## Static Public Member Functions

- `template<typename _E = Elt, enable_if_t<!is_same<_E, Givaro::Integer>::value> * = nullptr> static Residu cardinality ()`
- `template<typename _E = Elt, enable_if_t<is_same<_E, Givaro::Integer>::value> * = nullptr> static Residu cardinality ()`

## Protected Attributes

- `Field F`
- `const size_t m`
- `const size_t n`
- `const size_t iters`
- `const bool inplace`

## 12.17.1 Member Typedef Documentation

### 12.17.1.1 Field

```
template<typename Elt >
using Field = Modular<Elt>
```

### 12.17.1.2 Elt\_ptr

```
template<typename Elt >
using Elt_ptr = typename Field::Element_ptr
```

### 12.17.1.3 Residu

```
template<typename Elt >
using Residu = typename Field::Residu_t
```

### 12.17.1.4 enable\_if\_t

```
template<typename Elt >
template<bool B, class T = void>
using enable_if_t = typename std::enable_if<B, T>::type
```

### 12.17.1.5 is\_same\_element

```
template<typename Elt >
template<typename Simd >
using is_same_element = typename Simd::template is_same_element<Field>
```

### 12.17.1.6 enable\_if\_no\_simd\_t

```
template<typename Elt >
template<typename E >
using enable_if_no_simd_t = enable_if_t<Simd<E>::vect_size == 1>
```

### 12.17.1.7 enable\_if\_simd128\_t

```
template<typename Elt >
template<typename E >
using enable_if_simd128_t = enable_if_t<sizeof(E)*Simd<E>::vect_size == 16>
```

### 12.17.1.8 enable\_if\_simd256\_t

```
template<typename Elt >
template<typename E >
using enable_if_simd256_t = enable_if_t<sizeof(E)*Simd<E>::vect_size == 32>
```

### 12.17.1.9 enable\_if\_simd512\_t

```
template<typename Elt >
template<typename E >
using enable_if_simd512_t = enable_if_t<sizeof(E)*Simd<E>::vect_size == 64>
```

## 12.17.2 Constructor & Destructor Documentation

### 12.17.2.1 Bench()

```
template<typename Elt >
Bench (
 size_t m,
 size_t n,
 size_t iters,
 bool inplace) [inline]
```

## 12.17.3 Member Function Documentation

### 12.17.3.1 cardinality() [1/2]

```
template<typename Elt >
template<typename _E = Elt, enable_if_t<!is_same< _E, Givaro::Integer >::value > * = nullptr>
static Residu cardinality () [inline], [static]
```

### 12.17.3.2 cardinality() [2/2]

```
template<typename Elt >
template<typename _E = Elt, enable_if_t< is_same< _E, Givaro::Integer >::value > * = nullptr>
static Residu cardinality () [inline], [static]
```

### 12.17.3.3 doBenchs()

```
template<typename Elt >
template<typename Simd = NoSimd<Elt>, enable_if_t< is_same_element< Simd >::value > * =
nullptr>
void doBenchs () [inline]
```

### 12.17.3.4 run()

```
template<typename Elt >
template<typename _E = Elt, enable_if_t< is_same< _E, Elt >::value > * = nullptr, enable_if_no_simd_t<
_E > * = nullptr>
void run (
 bool allsimd) [inline]
```

## 12.17.4 Field Documentation

### 12.17.4.1 F

```
template<typename Elt >
Field F [protected]
```

### 12.17.4.2 m

```
template<typename Elt >
const size_t m [protected]
```

**12.17.4.3 n**

```
template<typename Elt >
const size_t n [protected]
```

**12.17.4.4 iters**

```
template<typename Elt >
const size_t iters [protected]
```

**12.17.4.5 inplace**

```
template<typename Elt >
const bool inplace [protected]
```

The documentation for this class was generated from the following file:

- [benchmark-storage-transpose.C](#)

**12.18 Bini Struct Reference**

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

**12.19 Block Struct Reference**

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

**12.20 BlockTransposeSIMD< Field, Simd, > Struct Template Reference**

```
#include <fflas_transpose.h>
```

**Public Member Functions**

- `template<size_t_s = Simd::vect_size, IsSimdSize<_s, 1> * = nullptr>`  
`void transpose (const Field &F, ConstElement_ptr A, size_t lda, Element_ptr B, size_t ldb) const`
- `template<size_t_s = Simd::vect_size, IsSimdSize<_s, 2> * = nullptr>`  
`void transpose (const Field &F, ConstElement_ptr A, size_t lda, Element_ptr B, size_t ldb) const`
- `template<size_t_s = Simd::vect_size, IsSimdSize<_s, 4> * = nullptr>`  
`void transpose (const Field &F, ConstElement_ptr A, size_t lda, Element_ptr B, size_t ldb) const`
- `template<size_t_s = Simd::vect_size, IsSimdSize<_s, 8> * = nullptr>`  
`void transpose (const Field &F, ConstElement_ptr A, size_t lda, Element_ptr B, size_t ldb) const`
- `template<size_t_s = Simd::vect_size, IsSimdSize<_s, 16> * = nullptr>`  
`void transpose (const Field &F, ConstElement_ptr A, size_t lda, Element_ptr B, size_t ldb) const`

**Static Public Member Functions**

- `static constexpr size_t size ()`
- `static const std::string info ()`

**12.20.1 Member Function Documentation****12.20.1.1 size()**

```
template<typename Field , typename Simd , typename std::enable_if< Simd::template is_same_↔
element< Field >::value >::type * = nullptr>
static constexpr size_t size () [inline], [static], [constexpr]
```

**12.20.1.2 info()**

```
template<typename Field , typename Simd , typename std::enable_if< Simd::template is_same_↵
element< Field >::value >::type * = nullptr>
static const std::string info () [inline], [static]
```

**12.20.1.3 transpose() [1/5]**

```
template<typename Field , typename Simd , typename std::enable_if< Simd::template is_same_↵
element< Field >::value >::type * = nullptr>
template<size_t _s = Simd::vect_size, IsSimdSize< _s, 1 > * = nullptr>
void transpose (
 const Field & F,
 ConstElement_ptr A,
 size_t lda,
 Element_ptr B,
 size_t ldb) const [inline]
```

**12.20.1.4 transpose() [2/5]**

```
template<typename Field , typename Simd , typename std::enable_if< Simd::template is_same_↵
element< Field >::value >::type * = nullptr>
template<size_t _s = Simd::vect_size, IsSimdSize< _s, 2 > * = nullptr>
void transpose (
 const Field & F,
 ConstElement_ptr A,
 size_t lda,
 Element_ptr B,
 size_t ldb) const [inline]
```

**12.20.1.5 transpose() [3/5]**

```
template<typename Field , typename Simd , typename std::enable_if< Simd::template is_same_↵
element< Field >::value >::type * = nullptr>
template<size_t _s = Simd::vect_size, IsSimdSize< _s, 4 > * = nullptr>
void transpose (
 const Field & F,
 ConstElement_ptr A,
 size_t lda,
 Element_ptr B,
 size_t ldb) const [inline]
```

**12.20.1.6 transpose() [4/5]**

```
template<typename Field , typename Simd , typename std::enable_if< Simd::template is_same_↵
element< Field >::value >::type * = nullptr>
template<size_t _s = Simd::vect_size, IsSimdSize< _s, 8 > * = nullptr>
void transpose (
 const Field & F,
 ConstElement_ptr A,
 size_t lda,
 Element_ptr B,
 size_t ldb) const [inline]
```

**12.20.1.7 transpose() [5/5]**

```
template<typename Field , typename Simd , typename std::enable_if< Simd::template is_same_↵
element< Field >::value >::type * = nullptr>
template<size_t _s = Simd::vect_size, IsSimdSize< _s, 16 > * = nullptr>
```

```
void transpose (
 const Field & F,
 ConstElement_ptr A,
 size_t lda,
 Element_ptr B,
 size_t ldb) const [inline]
```

The documentation for this struct was generated from the following file:

- [fflas\\_transpose.h](#)

## 12.21 callLUdivine\_small< Element > Class Template Reference

### Public Member Functions

- `template<class Field >`  
`size_t operator() (const Field &F, const FFLAS::FFLAS_DIAG Diag, const FFLAS::FFLAS_TRANSPOSE`  
`trans, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t`  
`*Q, const FFPACK::FFPACK_LU_TAG LuTag)`

### 12.21.1 Member Function Documentation

#### 12.21.1.1 operator>()

```
template<class Element >
template<class Field >
size_t operator() (
 const Field & F,
 const FFLAS::FFLAS_DIAG Diag,
 const FFLAS::FFLAS_TRANSPOSE trans,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t * P,
 size_t * Q,
 const FFPACK::FFPACK_LU_TAG LuTag) [inline]
```

The documentation for this class was generated from the following file:

- [ffpack\\_ludivine.inl](#)

## 12.22 callLUdivine\_small< double > Class Reference

### Public Member Functions

- `template<class Field >`  
`size_t operator() (const Field &F, const FFLAS::FFLAS_DIAG Diag, const FFLAS::FFLAS_TRANSPOSE`  
`trans, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t`  
`*Q, const FFPACK::FFPACK_LU_TAG LuTag)`

### 12.22.1 Member Function Documentation

#### 12.22.1.1 operator>()

```
template<class Field >
size_t operator() (
 const Field & F,
 const FFLAS::FFLAS_DIAG Diag,
 const FFLAS::FFLAS_TRANSPOSE trans,
 const size_t M,
```

```

const size_t N,
typename Field::Element_ptr A,
const size_t lda,
size_t * P,
size_t * Q,
const FFPACK::FFPACK_LU_TAG LuTag) [inline]

```

The documentation for this class was generated from the following file:

- [ffpack\\_ludivine.inl](#)

## 12.23 callLUdivine\_small< float > Class Reference

### Public Member Functions

- `template<class Field >`  
`size_t operator() (const Field &F, const FFLAS::FFLAS_DIAG Diag, const FFLAS::FFLAS_TRANSPOSE trans, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q, const FFPACK::FFPACK_LU_TAG LuTag)`

### 12.23.1 Member Function Documentation

#### 12.23.1.1 operator>()

```

template<class Field >
size_t operator() (
 const Field & F,
 const FFLAS::FFLAS_DIAG Diag,
 const FFLAS::FFLAS_TRANSPOSE trans,
 const size_t M,
 const size_t N,
 typename Field::Element_ptr A,
 const size_t lda,
 size_t * P,
 size_t * Q,
 const FFPACK::FFPACK_LU_TAG LuTag) [inline]

```

The documentation for this class was generated from the following file:

- [ffpack\\_ludivine.inl](#)

## 12.24 CharpolyFailed Class Reference

```
#include <ffpack.h>
```

The documentation for this class was generated from the following file:

- [ffpack.h](#)

## 12.25 Checker\_Empty< Field > Struct Template Reference

```
#include <checker_empty.h>
```

### Public Member Functions

- `template<typename... Params>`  
`Checker_Empty (Params... parameters)`
- `template<typename... Params>`  
`bool check (Params... parameters)`

## 12.25.1 Constructor & Destructor Documentation

### 12.25.1.1 Checker\_Empty()

```
template<class Field >
template<typename... Params>
Checker_Empty (
 Params... parameters) [inline]
```

## 12.25.2 Member Function Documentation

### 12.25.2.1 check()

```
template<class Field >
template<typename... Params>
bool check (
 Params... parameters) [inline]
```

The documentation for this struct was generated from the following file:

- [checker\\_empty.h](#)

## 12.26 CheckerImplem\_charpoly< Field, Polynomial > Class Template Reference

### Public Member Functions

- [CheckerImplem\\_charpoly](#) (const Field &F\_, const size\_t n\_, typename Field::ConstElement\_ptr A, size\_t lda\_)
- [CheckerImplem\\_charpoly](#) (typename Field::RandIter &G, const size\_t n\_, typename Field::ConstElement\_ptr A, size\_t lda\_)
- [~CheckerImplem\\_charpoly](#) ()
- [bool check](#) (Polynomial &g)

## 12.26.1 Constructor & Destructor Documentation

### 12.26.1.1 CheckerImplem\_charpoly() [1/2]

```
template<class Field , class Polynomial >
CheckerImplem_charpoly (
 const Field & F_,
 const size_t n_,
 typename Field::ConstElement_ptr A,
 size_t lda_) [inline]
```

### 12.26.1.2 CheckerImplem\_charpoly() [2/2]

```
template<class Field , class Polynomial >
CheckerImplem_charpoly (
 typename Field::RandIter & G,
 const size_t n_,
 typename Field::ConstElement_ptr A,
 size_t lda_) [inline]
```

### 12.26.1.3 ~CheckerImplem\_charpoly()

```
template<class Field , class Polynomial >
~CheckerImplem_charpoly () [inline]
```

## 12.26.2 Member Function Documentation

### 12.26.2.1 check()

```
template<class Field , class Polynomial >
bool check (
 Polynomial & g) [inline]
```

The documentation for this class was generated from the following file:

- [checker\\_charpoly.inl](#)

## 12.27 CheckerImplem\_charpoly< Givaro::ZRing< Givaro::Integer >, Polynomial > Class Template Reference

### Public Types

- `typedef Givaro::ZRing< Givaro::Integer > Ring`

### Public Member Functions

- `CheckerImplem_charpoly (const Ring &F_, const size_t n_, typename Ring::ConstElement_ptr A, size_t lda_)`
- `CheckerImplem_charpoly (typename Ring::RandIter &G, const size_t n_, typename Ring::ConstElement_ptr A, size_t lda_)`
- `~CheckerImplem_charpoly ()`
- `bool check (Polynomial &g)`

## 12.27.1 Member Typedef Documentation

### 12.27.1.1 Ring

```
template<class Polynomial >
typedef Givaro::ZRing<Givaro::Integer> Ring
```

## 12.27.2 Constructor & Destructor Documentation

### 12.27.2.1 CheckerImplem\_charpoly() [1/2]

```
template<class Polynomial >
CheckerImplem_charpoly (
 const Ring & F_,
 const size_t n_,
 typename Ring::ConstElement_ptr A,
 size_t lda_) [inline]
```

### 12.27.2.2 CheckerImplem\_charpoly() [2/2]

```
template<class Polynomial >
CheckerImplem_charpoly (
 typename Ring::RandIter & G,
 const size_t n_,
 typename Ring::ConstElement_ptr A,
 size_t lda_) [inline]
```

### 12.27.2.3 ~CheckerImplem\_charpoly()

```
template<class Polynomial >
~CheckerImplem_charpoly () [inline]
```



## 12.27.3 Member Function Documentation

### 12.27.3.1 check()

```
template<class Polynomial >
bool check (
 Polynomial & g) [inline]
```

The documentation for this class was generated from the following file:

- [checker\\_charpoly.inl](#)

## 12.28 CheckerImplem\_Det< Field > Class Template Reference

### Public Member Functions

- [CheckerImplem\\_Det \(const Field &F\\_, size\\_t n\\_, typename Field::ConstElement\\_ptr A, size\\_t lda\)](#)
- [CheckerImplem\\_Det \(typename Field::RandIter &G, size\\_t n\\_, typename Field::ConstElement\\_ptr A, size\\_t lda\)](#)
- [~CheckerImplem\\_Det \(\)](#)
- [bool check \(const typename Field::Element &det, typename Field::ConstElement\\_ptr LU, size\\_t lda, size\\_t \\*P, size\\_t \\*Q\) const](#)

*check if the Det factorization is correct.*

## 12.28.1 Constructor & Destructor Documentation

### 12.28.1.1 CheckerImplem\_Det() [1/2]

```
template<class Field >
CheckerImplem_Det (
 const Field & F_,
 size_t n_,
 typename Field::ConstElement_ptr A,
 size_t lda) [inline]
```

### 12.28.1.2 CheckerImplem\_Det() [2/2]

```
template<class Field >
CheckerImplem_Det (
 typename Field::RandIter & G,
 size_t n_,
 typename Field::ConstElement_ptr A,
 size_t lda) [inline]
```

### 12.28.1.3 ~CheckerImplem\_Det()

```
template<class Field >
~CheckerImplem_Det () [inline]
```

## 12.28.2 Member Function Documentation

### 12.28.2.1 check()

```
template<class Field >
bool check (
 const typename Field::Element & det,
 typename Field::ConstElement_ptr LU,
 size_t lda,
 size_t * P,
 size_t * Q) const [inline]
```

check if the Det factorization is correct.

Needs matrix in LU form

## Parameters

|                   |             |
|-------------------|-------------|
| <i>LU,storage</i> | for L and U |
| <i>det</i>        |             |
| <i>P</i>          |             |
| <i>Q</i>          |             |

The documentation for this class was generated from the following file:

- [checker\\_det.inl](#)

## 12.29 CheckerImplem\_fgemm< Field > Class Template Reference

### Public Member Functions

- [CheckerImplem\\_fgemm](#) (const Field &F\_, const size\_t m\_, const size\_t n\_, const size\_t k\_, const typename Field::Element beta, typename Field::Element\_ptr C, const size\_t ldc\_)
- [CheckerImplem\\_fgemm](#) (typename Field::RandIter &G, const size\_t m\_, const size\_t n\_, const size\_t k\_, const typename Field::Element beta, typename Field::Element\_ptr C, const size\_t ldc\_)
- [~CheckerImplem\\_fgemm](#) ()
- [bool check](#) (const FFLAS::FFLAS\_TRANSPOSE ta, const FFLAS::FFLAS\_TRANSPOSE tb, const typename Field::Element alpha, typename Field::ConstElement\_ptr A, const size\_t lda, typename Field::ConstElement\_ptr B, const size\_t ldb, typename Field::ConstElement\_ptr C)

### 12.29.1 Constructor & Destructor Documentation

#### 12.29.1.1 CheckerImplem\_fgemm() [1/2]

```
template<class Field >
CheckerImplem_fgemm (
 const Field & F_,
 const size_t m_,
 const size_t n_,
 const size_t k_,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc_) [inline]
```

#### 12.29.1.2 CheckerImplem\_fgemm() [2/2]

```
template<class Field >
CheckerImplem_fgemm (
 typename Field::RandIter & G,
 const size_t m_,
 const size_t n_,
 const size_t k_,
 const typename Field::Element beta,
 typename Field::Element_ptr C,
 const size_t ldc_) [inline]
```

#### 12.29.1.3 ~CheckerImplem\_fgemm()

```
template<class Field >
~CheckerImplem_fgemm () [inline]
```

## 12.29.2 Member Function Documentation

### 12.29.2.1 check()

```
template<class Field >
bool check (
 const FFLAS::FFLAS_TRANSPOSE ta,
 const FFLAS::FFLAS_TRANSPOSE tb,
 const typename Field::Element alpha,
 typename Field::ConstElement_ptr A,
 const size_t lda,
 typename Field::ConstElement_ptr B,
 const size_t ldb,
 typename Field::ConstElement_ptr C) [inline]
```

The documentation for this class was generated from the following file:

- [checker\\_fgemm.inl](#)

## 12.30 CheckerImplem\_ftrsm< Field > Class Template Reference

### Public Member Functions

- [CheckerImplem\\_ftrsm \(const Field &F\\_, const size\\_t m, const size\\_t n, const typename Field::Element alpha, const typename Field::ConstElement\\_ptr B, const size\\_t ldb\)](#)
- [CheckerImplem\\_ftrsm \(typename Field::RandIter &G, const size\\_t m, const size\\_t n, const typename Field::Element alpha, const typename Field::ConstElement\\_ptr B, const size\\_t ldb\)](#)
- [~CheckerImplem\\_ftrsm \(\)](#)
- [bool check \(const FFLAS::FFLAS\\_SIDE side, const FFLAS::FFLAS\\_UPLO uplo, const FFLAS::FFLAS\\_TRANSPOSE trans, const FFLAS::FFLAS\\_DIAG diag, const size\\_t m, const size\\_t n, typename Field::Element\\_ptr A, size\\_t lda, const typename Field::ConstElement\\_ptr X, size\\_t ldx\)](#)

### 12.30.1 Constructor & Destructor Documentation

#### 12.30.1.1 CheckerImplem\_ftrsm() [1/2]

```
template<class Field >
CheckerImplem_ftrsm (
 const Field & F_,
 const size_t m,
 const size_t n,
 const typename Field::Element alpha,
 const typename Field::ConstElement_ptr B,
 const size_t ldb) [inline]
```

#### 12.30.1.2 CheckerImplem\_ftrsm() [2/2]

```
template<class Field >
CheckerImplem_ftrsm (
 typename Field::RandIter & G,
 const size_t m,
 const size_t n,
 const typename Field::Element alpha,
 const typename Field::ConstElement_ptr B,
 const size_t ldb) [inline]
```

#### 12.30.1.3 ~CheckerImplem\_ftrsm()

```
template<class Field >
~CheckerImplem_ftrsm () [inline]
```

## 12.30.2 Member Function Documentation

### 12.30.2.1 check()

```
template<class Field >
bool check (
 const FFLAS::FFLAS_SIDE side,
 const FFLAS::FFLAS_UPLO uplo,
 const FFLAS::FFLAS_TRANSPOSE trans,
 const FFLAS::FFLAS_DIAG diag,
 const size_t m,
 const size_t n,
 typename Field::Element_ptr A,
 size_t lda,
 const typename Field::ConstElement_ptr X,
 size_t ldx) [inline]
```

The documentation for this class was generated from the following file:

- [checker\\_frsm.inl](#)

## 12.31 CheckerImplem\_invert< Field > Class Template Reference

### Public Member Functions

- [CheckerImplem\\_invert \(const Field &F\\_, const size\\_t m\\_, typename Field::ConstElement\\_ptr A, const size\\_t lda\\_\)](#)
- [CheckerImplem\\_invert \(typename Field::RandIter &G, const size\\_t m\\_, typename Field::ConstElement\\_ptr A, const size\\_t lda\\_\)](#)
- [~CheckerImplem\\_invert \(\)](#)
- [bool check \(typename Field::ConstElement\\_ptr A, int nullity\)](#)

### 12.31.1 Constructor & Destructor Documentation

#### 12.31.1.1 CheckerImplem\_invert() [1/2]

```
template<class Field >
CheckerImplem_invert (
 const Field & F_,
 const size_t m_,
 typename Field::ConstElement_ptr A,
 const size_t lda_) [inline]
```

#### 12.31.1.2 CheckerImplem\_invert() [2/2]

```
template<class Field >
CheckerImplem_invert (
 typename Field::RandIter & G,
 const size_t m_,
 typename Field::ConstElement_ptr A,
 const size_t lda_) [inline]
```

#### 12.31.1.3 ~CheckerImplem\_invert()

```
template<class Field >
~CheckerImplem_invert () [inline]
```

### 12.31.2 Member Function Documentation

#### 12.31.2.1 check()

```
template<class Field >
```

```
bool check (
 typename Field::ConstElement_ptr A,
 int nullity) [inline]
```

The documentation for this class was generated from the following file:

- [checker\\_invert.inl](#)

## 12.32 CheckerImplem\_PLUQ< Field > Class Template Reference

### Public Member Functions

- [CheckerImplem\\_PLUQ](#) (const Field &F\_, size\_t m\_, size\_t n\_, typename Field::ConstElement\_ptr A, size\_t lda)
- [CheckerImplem\\_PLUQ](#) (typename Field::RandIter &G, size\_t m\_, size\_t n\_, typename Field::ConstElement\_ptr A, size\_t lda)
- [~CheckerImplem\\_PLUQ](#) ()
- [bool check](#) (typename Field::ConstElement\_ptr A, size\_t lda, const FFLAS::FFLAS\_DIAG Diag, size\_t r, size\_t \*P, size\_t \*Q) const

*check if the PLUQ factorization is correct.*

### 12.32.1 Constructor & Destructor Documentation

#### 12.32.1.1 CheckerImplem\_PLUQ() [1/2]

```
template<class Field >
CheckerImplem_PLUQ (
 const Field & F_,
 size_t m_,
 size_t n_,
 typename Field::ConstElement_ptr A,
 size_t lda) [inline]
```

#### 12.32.1.2 CheckerImplem\_PLUQ() [2/2]

```
template<class Field >
CheckerImplem_PLUQ (
 typename Field::RandIter & G,
 size_t m_,
 size_t n_,
 typename Field::ConstElement_ptr A,
 size_t lda) [inline]
```

#### 12.32.1.3 ~CheckerImplem\_PLUQ()

```
template<class Field >
~CheckerImplem_PLUQ () [inline]
```

### 12.32.2 Member Function Documentation

#### 12.32.2.1 check()

```
template<class Field >
bool check (
 typename Field::ConstElement_ptr A,
 size_t lda,
 const FFLAS::FFLAS_DIAG Diag,
 size_t r,
 size_t * P,
 size_t * Q) const [inline]
```

check if the PLUQ factorization is correct.  
Returns true if  $w - P(L(U(Q.v))) == 0$

#### Parameters

|          |  |
|----------|--|
| <i>A</i> |  |
| <i>r</i> |  |
| <i>P</i> |  |
| <i>Q</i> |  |

The documentation for this class was generated from the following file:

- [checker\\_pluq.inl](#)

## 12.33 Classic Struct Reference

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

## 12.34 Column Struct Reference

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

## 12.35 CompactElement< Element > Struct Template Reference

### Public Types

- [typedef Element type](#)

### 12.35.1 Member Typedef Documentation

#### 12.35.1.1 type

```
template<class Element >
typedef Element type
```

The documentation for this struct was generated from the following file:

- [test-io.C](#)

## 12.36 CompactElement< double > Struct Reference

### Public Types

- [typedef int32\\_t type](#)

### 12.36.1 Member Typedef Documentation

#### 12.36.1.1 type

```
typedef int32_t type
```

The documentation for this struct was generated from the following file:

- [test-io.C](#)

## 12.37 CompactElement< float > Struct Reference

### Public Types

- [typedef int16\\_t type](#)

### 12.37.1 Member Typedef Documentation

#### 12.37.1.1 type

[typedef int16\\_t type](#)

The documentation for this struct was generated from the following file:

- [test-io.C](#)

## 12.38 CompactElement< int16\_t > Struct Reference

### Public Types

- [typedef int8\\_t type](#)

### 12.38.1 Member Typedef Documentation

#### 12.38.1.1 type

[typedef int8\\_t type](#)

The documentation for this struct was generated from the following file:

- [test-io.C](#)

## 12.39 CompactElement< int32\_t > Struct Reference

### Public Types

- [typedef int16\\_t type](#)

### 12.39.1 Member Typedef Documentation

#### 12.39.1.1 type

[typedef int16\\_t type](#)

The documentation for this struct was generated from the following file:

- [test-io.C](#)

## 12.40 CompactElement< int64\_t > Struct Reference

### Public Types

- [typedef int32\\_t type](#)

### 12.40.1 Member Typedef Documentation

#### 12.40.1.1 type

[typedef int32\\_t type](#)

The documentation for this struct was generated from the following file:

- [test-io.C](#)



## 12.41 compatible\_data\_type< Field > Struct Template Reference

### Static Public Attributes

- [static constexpr bool value = true](#)

### 12.41.1 Field Documentation

#### 12.41.1.1 value

```
template<typename Field >
constexpr bool value = true [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [benchmark-fgemv.C](#)

## 12.42 compatible\_data\_type< Givaro::ZRing< double > > Struct Reference

### Static Public Attributes

- [static constexpr bool value = false](#)

### 12.42.1 Field Documentation

#### 12.42.1.1 value

```
constexpr bool value = false [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [benchmark-fgemv.C](#)

## 12.43 compatible\_data\_type< Givaro::ZRing< float > > Struct Reference

### Static Public Attributes

- [static constexpr bool value = false](#)

### 12.43.1 Field Documentation

#### 12.43.1.1 value

```
constexpr bool value = false [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [benchmark-fgemv.C](#)

## 12.44 Compose< H1, H2 > Struct Template Reference

### Public Member Functions

- [Compose \(\)](#)
- [Compose \(const Compose &other\)](#)
- [Compose \(const Sequential &S\)](#)
- [Compose \(size\\_t th1, size\\_t th2\)](#)
- [Compose \(const H1 &o1, const H2 &o2\)](#)
- [H1 first\\_component \(\) const](#)
- [H2 second\\_component \(\) const](#)

## Friends

- `std::ostream & operator<< (std::ostream &o, const Compose &c)`

## 12.44.1 Constructor & Destructor Documentation

### 12.44.1.1 Compose() [1/5]

```
template<typename H1 = Sequential, typename H2 = Sequential>
Compose () [inline]
```

### 12.44.1.2 Compose() [2/5]

```
template<typename H1 = Sequential, typename H2 = Sequential>
Compose (
 const Compose< H1, H2 > & other) [inline]
```

### 12.44.1.3 Compose() [3/5]

```
template<typename H1 = Sequential, typename H2 = Sequential>
Compose (
 const Sequential & S) [inline]
```

### 12.44.1.4 Compose() [4/5]

```
template<typename H1 = Sequential, typename H2 = Sequential>
Compose (
 size_t th1,
 size_t th2) [inline]
```

### 12.44.1.5 Compose() [5/5]

```
template<typename H1 = Sequential, typename H2 = Sequential>
Compose (
 const H1 & o1,
 const H2 & o2) [inline]
```

## 12.44.2 Member Function Documentation

### 12.44.2.1 first\_component()

```
template<typename H1 = Sequential, typename H2 = Sequential>
H1 first_component () const [inline]
```

### 12.44.2.2 second\_component()

```
template<typename H1 = Sequential, typename H2 = Sequential>
H2 second_component () const [inline]
```

## 12.44.3 Friends And Related Symbol Documentation

### 12.44.3.1 operator<<

```
template<typename H1 = Sequential, typename H2 = Sequential>
std::ostream & operator<< (
 std::ostream & o,
 const Compose< H1, H2 > & c) [friend]
```

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

## 12.45 Simd128\_impl< true, true, false, 2 >::Converter Union Reference

### Data Fields

- [vect\\_tv](#)
- [scalar\\_tt](#) [vect\_size]

### 12.45.1 Field Documentation

#### 12.45.1.1 v

[vect\\_tv](#) v

#### 12.45.1.2 t

[scalar\\_tt](#) t [vect\_size]

The documentation for this union was generated from the following file:

- [simd128\\_int16.inl](#)

## 12.46 Simd128\_impl< true, true, false, 4 >::Converter Union Reference

### Data Fields

- [vect\\_tv](#)
- [scalar\\_tt](#) [vect\_size]

### 12.46.1 Field Documentation

#### 12.46.1.1 v

[vect\\_tv](#) v

#### 12.46.1.2 t

[scalar\\_tt](#) t [vect\_size]

The documentation for this union was generated from the following file:

- [simd128\\_int32.inl](#)

## 12.47 Simd128\_impl< true, true, false, 8 >::Converter Union Reference

### Data Fields

- [vect\\_tv](#)
- [scalar\\_tt](#) [vect\_size]

### 12.47.1 Field Documentation

#### 12.47.1.1 v

[vect\\_tv](#) v

#### 12.47.1.2 t

[scalar\\_tt](#) t [vect\_size]

The documentation for this union was generated from the following file:

- [simd128\\_int64.inl](#)

## 12.48 Simd128\_impl< true, true, true, 2 >::Converter Union Reference

### Data Fields

- [vect\\_tv](#)
- [scalar\\_tt](#) [vect\_size]

### 12.48.1 Field Documentation

#### 12.48.1.1 v

[vect\\_tv](#) v

#### 12.48.1.2 t

[scalar\\_tt](#) t [vect\_size]

The documentation for this union was generated from the following file:

- [simd128\\_int16.inl](#)

## 12.49 Simd128\_impl< true, true, true, 4 >::Converter Union Reference

### Data Fields

- [vect\\_tv](#)
- [scalar\\_tt](#) [vect\_size]

### 12.49.1 Field Documentation

#### 12.49.1.1 v

[vect\\_tv](#) v

#### 12.49.1.2 t

[scalar\\_tt](#) t [vect\_size]

The documentation for this union was generated from the following file:

- [simd128\\_int32.inl](#)

## 12.50 Simd128\_impl< true, true, true, 8 >::Converter Union Reference

### Data Fields

- [vect\\_tv](#)
- [scalar\\_tt](#) [vect\_size]

### 12.50.1 Field Documentation

#### 12.50.1.1 v

[vect\\_tv](#) v

#### 12.50.1.2 t

[scalar\\_tt](#) t [vect\_size]

The documentation for this union was generated from the following file:

- [simd128\\_int64.inl](#)

## 12.51 Simd256\_impl< true, false, true, 8 >::Converter Union Reference

### Data Fields

- [vect\\_tv](#)
- [scalar\\_tt](#) [vect\_size]

### 12.51.1 Field Documentation

#### 12.51.1.1 v

[vect\\_tv](#) v

#### 12.51.1.2 t

[scalar\\_tt](#) t [vect\_size]

The documentation for this union was generated from the following file:

- [simd256\\_double.inl](#)

## 12.52 Simd256\_impl< true, true, false, 2 >::Converter Union Reference

### Data Fields

- [vect\\_tv](#)
- [scalar\\_tt](#) [vect\_size]

### 12.52.1 Field Documentation

#### 12.52.1.1 v

[vect\\_tv](#) v

#### 12.52.1.2 t

[scalar\\_tt](#) t [vect\_size]

The documentation for this union was generated from the following file:

- [simd256\\_int16.inl](#)

## 12.53 Simd256\_impl< true, true, false, 4 >::Converter Union Reference

### Data Fields

- [vect\\_tv](#)
- [scalar\\_tt](#) [vect\_size]

### 12.53.1 Field Documentation

#### 12.53.1.1 v

[vect\\_tv](#) v

#### 12.53.1.2 t

[scalar\\_tt](#) t

The documentation for this union was generated from the following files:

- [simd256\\_int32.inl](#)
- [simd512\\_int32.inl](#)

## 12.54 Simd256\_impl< true, true, false, 8 >::Converter Union Reference

### Data Fields

- [vect\\_tv](#)
- [scalar\\_tt](#) [vect\_size]

### 12.54.1 Field Documentation

#### 12.54.1.1 v

[vect\\_tv](#) v

#### 12.54.1.2 t

[scalar\\_tt](#) t [vect\_size]

The documentation for this union was generated from the following file:

- [simd256\\_int64.inl](#)

## 12.55 Simd256\_impl< true, true, true, 2 >::Converter Union Reference

### Data Fields

- [vect\\_tv](#)
- [scalar\\_tt](#) [vect\_size]

### 12.55.1 Field Documentation

#### 12.55.1.1 v

[vect\\_tv](#) v

#### 12.55.1.2 t

[scalar\\_tt](#) t [vect\_size]

The documentation for this union was generated from the following file:

- [simd256\\_int16.inl](#)

## 12.56 Simd256\_impl< true, true, true, 4 >::Converter Union Reference

### Data Fields

- [vect\\_tv](#)
- [scalar\\_tt](#) [vect\_size]

### 12.56.1 Field Documentation

#### 12.56.1.1 v

[vect\\_tv](#) v

#### 12.56.1.2 t

[scalar\\_tt](#) t

The documentation for this union was generated from the following files:

- [simd256\\_int32.inl](#)
- [simd512\\_int32.inl](#)

## 12.57 Simd256\_impl< true, true, true, 8 >::Converter Union Reference

### Data Fields

- [vect\\_t v](#)
- [scalar\\_t t](#) [vect\_size]

### 12.57.1 Field Documentation

#### 12.57.1.1 v

[vect\\_t v](#)

#### 12.57.1.2 t

[scalar\\_t t](#) [vect\_size]

The documentation for this union was generated from the following file:

- [simd256\\_int64.inl](#)

## 12.58 Simd512\_impl< true, true, false, 8 >::Converter Union Reference

### Data Fields

- [vect\\_t v](#)
- [scalar\\_t t](#) [vect\_size]

### 12.58.1 Field Documentation

#### 12.58.1.1 v

[vect\\_t v](#)

#### 12.58.1.2 t

[scalar\\_t t](#) [vect\_size]

The documentation for this union was generated from the following file:

- [simd512\\_int64.inl](#)

## 12.59 Simd512\_impl< true, true, true, 8 >::Converter Union Reference

### Data Fields

- [vect\\_t v](#)
- [scalar\\_t t](#) [vect\_size]

### 12.59.1 Field Documentation

#### 12.59.1.1 v

[vect\\_t v](#)

#### 12.59.1.2 t

[scalar\\_t t](#) [vect\_size]

The documentation for this union was generated from the following file:

- [simd512\\_int64.inl](#)

## 12.60 ConvertTo< T > Struct Template Reference

Force conversion to appropriate element type of ElementCategory T.

```
#include <field-traits.h>
```

### 12.60.1 Detailed Description

```
template<class T>
```

```
struct FFLAS::ModeCategories::ConvertTo< T >
```

Force conversion to appropriate element type of ElementCategory T.

e.g.

- ConvertTo<ElementCategories::MachineFloatTag> tries conversion of Modular<int> to Modular<double>
- ConvertTo<ElementCategories::FixedPreIntTag> tries conversion of Modular<Integer> to Modular<RecInt<K>>
- ConvertTo<ElementCategories::ArbitraryPreIntTag> tries conversion of Modular<Integer> to RNSInteger

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.61 Coo< ValT, IdxT > Struct Template Reference

### Public Types

- `using Self = Coo< ValT, IdxT >`

### Public Member Functions

- `Coo (ValT v, IdxT r, IdxT c)`
- `Coo ()=default`
- `Coo (const Self &)=default`
- `Coo (Self &&)=default`
- `Self & operator= (const Self &)=default`
- `Self & operator= (Self &&)=default`

### Data Fields

- `ValT val = 0`
- `IdxT row = 0`
- `IdxT col = 0`

### 12.61.1 Member Typedef Documentation

#### 12.61.1.1 Self

```
template<class ValT , class IdxT >
using Self = Coo<ValT, IdxT>
```

### 12.61.2 Constructor & Destructor Documentation

#### 12.61.2.1 Coo() [1/4]

```
template<class ValT , class IdxT >
Coo (
 ValT v,
 IdxT r,
 IdxT c) [inline]
```



**12.61.2.2 `Coo()` [2/4]**

```
template<class ValT , class IdxT >
Coo () [default]
```

**12.61.2.3 `Coo()` [3/4]**

```
template<class ValT , class IdxT >
Coo (
 const Self &) [default]
```

**12.61.2.4 `Coo()` [4/4]**

```
template<class ValT , class IdxT >
Coo (
 Self &&) [default]
```

**12.61.3 Member Function Documentation****12.61.3.1 `operator=()` [1/2]**

```
template<class ValT , class IdxT >
Self & operator= (
 const Self &) [default]
```

**12.61.3.2 `operator=()` [2/2]**

```
template<class ValT , class IdxT >
Self & operator= (
 Self &&) [default]
```

**12.61.4 Field Documentation****12.61.4.1 `val`**

```
template<class ValT , class IdxT >
ValT val = 0
```

**12.61.4.2 `row`**

```
template<class ValT , class IdxT >
IdxT row = 0
```

**12.61.4.3 `col`**

```
template<class ValT , class IdxT >
IdxT col = 0
```

The documentation for this struct was generated from the following file:

- [csr\\_hyb\\_utils.inl](#)

**12.62 `Coo< Field > Struct` Template Reference**

```
#include <read_sparse.h>
```

**Public Member Functions**

- `Coo()`=default
- `Coo` (typename `Field::Element` v, `index_t` r, `index_t` c)
- `Coo` (const `Self` &)=default

- `Coo (Self &&)=default`
- `Self & operator= (const Self &)=default`
- `Self & operator= (Self &&)=default`

### Data Fields

- `Field::Element val = 0`
- `index_t col = 0`
- `index_t row = 0`
- `bool deleted = false`

## 12.62.1 Constructor & Destructor Documentation

### 12.62.1.1 Coo() [1/4]

```
template<class Field >
Coo () [default]
```

### 12.62.1.2 Coo() [2/4]

```
template<class Field >
Coo (
 typename Field::Element v,
 index_t r,
 index_t c) [inline]
```

### 12.62.1.3 Coo() [3/4]

```
template<class Field >
Coo (
 const Self &) [default]
```

### 12.62.1.4 Coo() [4/4]

```
template<class Field >
Coo (
 Self &&) [default]
```

## 12.62.2 Member Function Documentation

### 12.62.2.1 operator=() [1/2]

```
template<class Field >
Self & operator= (
 const Self &) [default]
```

### 12.62.2.2 operator=() [2/2]

```
template<class Field >
Self & operator= (
 Self &&) [default]
```

## 12.62.3 Field Documentation

### 12.62.3.1 val

```
template<class Field >
Field::Element val = 0
```

**12.62.3.2 col**

```
template<class Field >
index_t col = 0
```

**12.62.3.3 row**

```
template<class Field >
index_t row = 0
```

**12.62.3.4 deleted**

```
template<class Field >
bool deleted = false
```

The documentation for this struct was generated from the following file:

- [read\\_sparse.h](#)

**12.63 `Coo< ValT, IdxT >` Struct Template Reference****Public Types**

- `using Self = Coo< ValT, IdxT >`

**Public Member Functions**

- `Coo (ValT v, IdxT r, IdxT c)`
- `Coo ()=default`
- `Coo (const Self &)=default`
- `Coo (Self &&)=default`
- `Self & operator= (const Self &)=default`
- `Self & operator= (Self &&)=default`

**Data Fields**

- `ValT val = 0`
- `IdxT row = 0`
- `IdxT col = 0`

**12.63.1 Member Typedef Documentation****12.63.1.1 Self**

```
template<class ValT , class IdxT >
using Self = Coo<ValT, IdxT>
```

**12.63.2 Constructor & Destructor Documentation****12.63.2.1 `Coo()` [1/4]**

```
template<class ValT , class IdxT >
Coo (
 ValT v,
 IdxT r,
 IdxT c) [inline]
```

**12.63.2.2 `Coo()` [2/4]**

```
template<class ValT , class IdxT >
Coo () [default]
```

**12.63.2.3** **Coo()** [3/4]

```
template<class ValT , class IdxT >
Coo (
 const Self &) [default]
```

**12.63.2.4** **Coo()** [4/4]

```
template<class ValT , class IdxT >
Coo (
 Self &&) [default]
```

**12.63.3** **Member Function Documentation****12.63.3.1** **operator=()** [1/2]

```
template<class ValT , class IdxT >
Self & operator= (
 const Self &) [default]
```

**12.63.3.2** **operator=()** [2/2]

```
template<class ValT , class IdxT >
Self & operator= (
 Self &&) [default]
```

**12.63.4** **Field Documentation****12.63.4.1** **val**

```
template<class ValT , class IdxT >
ValT val = 0
```

**12.63.4.2** **row**

```
template<class ValT , class IdxT >
IdxT row = 0
```

**12.63.4.3** **col**

```
template<class ValT , class IdxT >
IdxT col = 0
```

The documentation for this struct was generated from the following file:

- [sell\\_utils.inl](#)

**12.64** **CooMat< Field > Struct Template Reference**

```
#include <fflas_sparse.h>
```

**Data Fields**

- [FFLAS::Sparse< Field, SparseMatrix\\_t::COO, int16\\_t > \\* \\_coo16 = nullptr](#)
- [FFLAS::Sparse< Field, SparseMatrix\\_t::COO, int32\\_t > \\* \\_coo32 = nullptr](#)
- [FFLAS::Sparse< Field, SparseMatrix\\_t::COO, int64\\_t > \\* \\_coo64 = nullptr](#)
- [FFLAS::Sparse< Field, SparseMatrix\\_t::COO\\_ZO, int16\\_t > \\* \\_coo16\\_zo = nullptr](#)
- [FFLAS::Sparse< Field, SparseMatrix\\_t::COO\\_ZO, int32\\_t > \\* \\_coo32\\_zo = nullptr](#)
- [FFLAS::Sparse< Field, SparseMatrix\\_t::COO\\_ZO, int64\\_t > \\* \\_coo64\\_zo = nullptr](#)

### 12.64.1 Field Documentation

#### 12.64.1.1 \_coo16

```
template<class Field >
FFLAS::Sparse<Field, SparseMatrix_t::COO, int16_t>* _coo16 = nullptr
```

#### 12.64.1.2 \_coo32

```
template<class Field >
FFLAS::Sparse<Field, SparseMatrix_t::COO, int32_t>* _coo32 = nullptr
```

#### 12.64.1.3 \_coo64

```
template<class Field >
FFLAS::Sparse<Field, SparseMatrix_t::COO, int64_t>* _coo64 = nullptr
```

#### 12.64.1.4 \_coo16\_zo

```
template<class Field >
FFLAS::Sparse<Field, SparseMatrix_t::COO_ZO, int16_t>* _coo16_zo = nullptr
```

#### 12.64.1.5 \_coo32\_zo

```
template<class Field >
FFLAS::Sparse<Field, SparseMatrix_t::COO_ZO, int32_t>* _coo32_zo = nullptr
```

#### 12.64.1.6 \_coo64\_zo

```
template<class Field >
FFLAS::Sparse<Field, SparseMatrix_t::COO_ZO, int64_t>* _coo64_zo = nullptr
```

The documentation for this struct was generated from the following file:

- [fflas\\_sparse.h](#)

## 12.65 count\_nonconst\_lvalue\_reference< T > Struct Template Reference

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.66 count\_nonconst\_lvalue\_reference< const T &, O... > Struct Template Reference

### Static Public Attributes

- `static constexpr size_t n = count_nonconst_lvalue_reference<O...>::n`

### 12.66.1 Field Documentation

#### 12.66.1.1 n

```
template<typename T , typename... O>
constexpr size_t n = count_nonconst_lvalue_reference<O...>::n [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.67 `count_nonconst_lvalue_reference< T &, O... >` Struct Template Reference

### Static Public Attributes

- [static constexpr size\\_t n](#)

### 12.67.1 Field Documentation

#### 12.67.1.1 `n`

```
template<typename T , typename... O>
constexpr size_t n [static], [constexpr]
Initial value:
= std::integral_constant<size_t, 1>::value
 + count_nonconst_lvalue_reference<O...>::n
```

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.68 `count_nonconst_lvalue_reference< T, O... >` Struct Template Reference

### Static Public Attributes

- [static constexpr size\\_t n = count\\_nonconst\\_lvalue\\_reference<O...>::n](#)

### 12.68.1 Field Documentation

#### 12.68.1.1 `n`

```
template<typename T , typename... O>
constexpr size_t n = count_nonconst_lvalue_reference<O...>::n [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.69 `count_nonconst_lvalue_reference<>` Struct Reference

### Static Public Attributes

- [static constexpr size\\_t n = std::integral\\_constant<size\\_t, 0>::value](#)

### 12.69.1 Field Documentation

#### 12.69.1.1 `n`

```
constexpr size_t n = std::integral_constant<size_t, 0>::value [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.70 `CsrMat< Field >` Struct Template Reference

```
#include <fflas_sparse.h>
```

**Data Fields**

- [FFLAS::Sparse<Field, SparseMatrix\\_t::CSR, int16\\_t> \\* \\_csr16 = nullptr](#)
- [FFLAS::Sparse<Field, SparseMatrix\\_t::CSR, int32\\_t> \\* \\_csr32 = nullptr](#)
- [FFLAS::Sparse<Field, SparseMatrix\\_t::CSR, int64\\_t> \\* \\_csr64 = nullptr](#)
- [FFLAS::Sparse<Field, SparseMatrix\\_t::CSR\\_ZO, int16\\_t> \\* \\_csr16\\_zo = nullptr](#)
- [FFLAS::Sparse<Field, SparseMatrix\\_t::CSR\\_ZO, int32\\_t> \\* \\_csr32\\_zo = nullptr](#)
- [FFLAS::Sparse<Field, SparseMatrix\\_t::CSR\\_ZO, int64\\_t> \\* \\_csr64\\_zo = nullptr](#)

**12.70.1 Field Documentation****12.70.1.1 \_csr16**

```
template<class Field >
FFLAS::Sparse<Field, SparseMatrix_t::CSR, int16_t>* _csr16 = nullptr
```

**12.70.1.2 \_csr32**

```
template<class Field >
FFLAS::Sparse<Field, SparseMatrix_t::CSR, int32_t>* _csr32 = nullptr
```

**12.70.1.3 \_csr64**

```
template<class Field >
FFLAS::Sparse<Field, SparseMatrix_t::CSR, int64_t>* _csr64 = nullptr
```

**12.70.1.4 \_csr16\_zo**

```
template<class Field >
FFLAS::Sparse<Field, SparseMatrix_t::CSR_ZO, int16_t>* _csr16_zo = nullptr
```

**12.70.1.5 \_csr32\_zo**

```
template<class Field >
FFLAS::Sparse<Field, SparseMatrix_t::CSR_ZO, int32_t>* _csr32_zo = nullptr
```

**12.70.1.6 \_csr64\_zo**

```
template<class Field >
FFLAS::Sparse<Field, SparseMatrix_t::CSR_ZO, int64_t>* _csr64_zo = nullptr
```

The documentation for this struct was generated from the following file:

- [fflas\\_sparse.h](#)

**12.71 DefaultBoundedTag Struct Reference**

Use standard field operations, but keeps track of bounds on input and output.

```
#include <field-traits.h>
```

**12.71.1 Detailed Description**

Use standard field operations, but keeps track of bounds on input and output.

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

**12.72 DefaultTag Struct Reference**

No specific mode of action: use standard field operations.

```
#include <field-traits.h>
```

### 12.72.1 Detailed Description

No specific mode of action: use standard field operations.

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.73 DelayedTag Struct Reference

Performs field operations with delayed mod reductions. Ensures result is reduced.

```
#include <field-traits.h>
```

### 12.73.1 Detailed Description

Performs field operations with delayed mod reductions. Ensures result is reduced.

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.74 DivideAndConquer Struct Reference

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

## 12.75 ElementTraits< Element > Struct Template Reference

[ElementTraits](#).

```
#include <field-traits.h>
```

### Public Types

- [typedef ElementCategories::GenericTag value](#)

### 12.75.1 Detailed Description

```
template<class Element>
```

```
struct FFLAS::ElementTraits< Element >
```

[ElementTraits](#).

### 12.75.2 Member Typedef Documentation

#### 12.75.2.1 value

```
template<class Element >
```

```
typedef ElementCategories::GenericTag value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.76 ElementTraits< double > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef ElementCategories::MachineFloatTag value](#)



## 12.76.1 Member Typedef Documentation

### 12.76.1.1 value

`typedef ElementCategories::MachineFloatTag value`

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.77 ElementTraits< FFPACK::rns\_double\_elt > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- `typedef ElementCategories::RNSElementTag value`

## 12.77.1 Member Typedef Documentation

### 12.77.1.1 value

`typedef ElementCategories::RNSElementTag value`

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.78 ElementTraits< float > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- `typedef ElementCategories::MachineFloatTag value`

## 12.78.1 Member Typedef Documentation

### 12.78.1.1 value

`typedef ElementCategories::MachineFloatTag value`

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.79 ElementTraits< Givaro::Integer > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- `typedef ElementCategories::ArbitraryPrecIntTag value`

## 12.79.1 Member Typedef Documentation

### 12.79.1.1 value

`typedef ElementCategories::ArbitraryPrecIntTag value`

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.80 ElementTraits< int16\_t > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef ElementCategories::MachineIntTag value](#)

### 12.80.1 Member Typedef Documentation

#### 12.80.1.1 value

```
typedef ElementCategories::MachineIntTag value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.81 ElementTraits< int32\_t > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef ElementCategories::MachineIntTag value](#)

### 12.81.1 Member Typedef Documentation

#### 12.81.1.1 value

```
typedef ElementCategories::MachineIntTag value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.82 ElementTraits< int64\_t > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef ElementCategories::MachineIntTag value](#)

### 12.82.1 Member Typedef Documentation

#### 12.82.1.1 value

```
typedef ElementCategories::MachineIntTag value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.83 ElementTraits< int8\_t > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef ElementCategories::MachineIntTag value](#)

### 12.83.1 Member Typedef Documentation

#### 12.83.1.1 value

`typedef ElementCategories::MachineIntTag value`

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.84 ElementTraits< RecInt::rint< K > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef ElementCategories::FixedPrecIntTag value](#)

### 12.84.1 Member Typedef Documentation

#### 12.84.1.1 value

`template<size_t K>`

`typedef ElementCategories::FixedPrecIntTag value`

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.85 ElementTraits< RecInt::rmint< K, MG > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef ElementCategories::FixedPrecIntTag value](#)

### 12.85.1 Member Typedef Documentation

#### 12.85.1.1 value

`template<size_t K, int MG>`

`typedef ElementCategories::FixedPrecIntTag value`

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.86 ElementTraits< RecInt::ruint< K > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef ElementCategories::FixedPrecIntTag value](#)

## 12.86.1 Member Typedef Documentation

### 12.86.1.1 value

```
template<size_t K>
```

```
typedef ElementCategories::FixedPrecIntTag value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.87 ElementTraits< uint16\_t > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef ElementCategories::MachineIntTag value](#)

## 12.87.1 Member Typedef Documentation

### 12.87.1.1 value

```
typedef ElementCategories::MachineIntTag value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.88 ElementTraits< uint32\_t > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef ElementCategories::MachineIntTag value](#)

## 12.88.1 Member Typedef Documentation

### 12.88.1.1 value

```
typedef ElementCategories::MachineIntTag value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.89 ElementTraits< uint64\_t > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef ElementCategories::MachineIntTag value](#)

## 12.89.1 Member Typedef Documentation

### 12.89.1.1 value

```
typedef ElementCategories::MachineIntTag value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.90 ElementTraits< uint8\_t > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef ElementCategories::MachineIntTag value](#)

### 12.90.1 Member Typedef Documentation

#### 12.90.1.1 value

```
typedef ElementCategories::MachineIntTag value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.91 EIMat< Field > Struct Template Reference

```
#include <fflas_sparse.h>
```

### Data Fields

- [FFLAS::Sparse< Field, SparseMatrix\\_t::ELL, int16\\_t > \\* \\_ell16 = nullptr](#)
- [FFLAS::Sparse< Field, SparseMatrix\\_t::ELL, int32\\_t > \\* \\_ell32 = nullptr](#)
- [FFLAS::Sparse< Field, SparseMatrix\\_t::ELL, int64\\_t > \\* \\_ell64 = nullptr](#)
- [FFLAS::Sparse< Field, SparseMatrix\\_t::ELL\\_ZO, int16\\_t > \\* \\_ell16\\_zo = nullptr](#)
- [FFLAS::Sparse< Field, SparseMatrix\\_t::ELL\\_ZO, int32\\_t > \\* \\_ell32\\_zo = nullptr](#)
- [FFLAS::Sparse< Field, SparseMatrix\\_t::ELL\\_ZO, int64\\_t > \\* \\_ell64\\_zo = nullptr](#)

### 12.91.1 Field Documentation

#### 12.91.1.1 \_ell16

```
template<class Field >
FFLAS::Sparse<Field, SparseMatrix_t::ELL, int16_t>* _ell16 = nullptr
```

#### 12.91.1.2 \_ell32

```
template<class Field >
FFLAS::Sparse<Field, SparseMatrix_t::ELL, int32_t>* _ell32 = nullptr
```

#### 12.91.1.3 \_ell64

```
template<class Field >
FFLAS::Sparse<Field, SparseMatrix_t::ELL, int64_t>* _ell64 = nullptr
```

#### 12.91.1.4 \_ell16\_zo

```
template<class Field >
FFLAS::Sparse<Field, SparseMatrix_t::ELL_ZO, int16_t>* _ell16_zo = nullptr
```

#### 12.91.1.5 \_ell32\_zo

```
template<class Field >
FFLAS::Sparse<Field, SparseMatrix_t::ELL_ZO, int32_t>* _ell32_zo = nullptr
```

### 12.91.1.6 `_ell64_zo`

```
template<class Field >
FFLAS::Sparse<Field, SparseMatrix_t::ELL_ZO, int64_t>* _ell64_zo = nullptr
```

The documentation for this struct was generated from the following file:

- [fflas\\_sparse.h](#)

## 12.92 Failure Class Reference

A precondition failed.

```
#include <debug.h>
```

### Public Member Functions

- [Failure](#) ()
- [void operator\(\)](#) ([const char](#) \*function, [int](#) line, [const char](#) \*check)
- [void operator\(\)](#) ([const char](#) \*function, [const char](#) \*file, [int](#) line, [const char](#) \*check)
- [void setErrorStream](#) ([std::ostream](#) &stream)
- [std::ostream](#) & [print](#) ([std::ostream](#) &o) [const](#)

### Protected Attributes

- [std::ostream](#) \* [\\_errorStream](#)

### 12.92.1 Detailed Description

A precondition failed.

The `throw` mechanism is usually used here as in

```
if (!check)
failure()(__func__, __LINE__, "this check just failed");
```

The parameters of the constructor help debugging.

### 12.92.2 Constructor & Destructor Documentation

#### 12.92.2.1 `Failure()`

```
Failure () [inline]
```

### 12.92.3 Member Function Documentation

#### 12.92.3.1 `operator>()` [1/2]

```
void operator() (
 const char * function,
 int line,
 const char * check) [inline]
```

A precondition failed.

#### Parameters

|                 |                                                                   |
|-----------------|-------------------------------------------------------------------|
| <i>function</i> | usually <code>__func__</code> , the function that threw the error |
| <i>line</i>     | usually <code>__LINE__</code> , the line where it happened        |
| <i>check</i>    | a string telling what failed.                                     |

#### 12.92.3.2 `operator>()` [2/2]

```
void operator() (
```

```

const char * function,
const char * file,
int line,
const char * check) [inline]

```

A precondition failed. The parameter help debugging. This is not much different from the previous except we can dig faster in the file where the exception was triggered.

#### Parameters

|                 |                                                                   |
|-----------------|-------------------------------------------------------------------|
| <i>function</i> | usually <code>__func__</code> , the function that threw the error |
| <i>file</i>     | usually <code>__FILE__</code> , the file where this function is   |
| <i>line</i>     | usually <code>__LINE__</code> , the line where it happened        |
| <i>check</i>    | a string telling what failed.                                     |

### 12.92.3.3 setErrorStream()

```

void setErrorStream (
 std::ostream & stream)

```

### 12.92.3.4 print()

```

std::ostream & print (
 std::ostream & o) const [inline]

```

overload the virtual print of LinboxError.

#### Parameters

|          |               |
|----------|---------------|
| <i>o</i> | output stream |
|----------|---------------|

## 12.92.4 Field Documentation

### 12.92.4.1 \_errorStream

```
std::ostream* _errorStream [protected]
```

The documentation for this class was generated from the following file:

- [debug.h](#)

## 12.93 FailureCharpolyCheck Class Reference

```
#include <checkers_ffpack.h>
```

The documentation for this class was generated from the following file:

- [checkers\\_ffpack.h](#)

## 12.94 FailureDetCheck Class Reference

```
#include <checkers_ffpack.h>
```

The documentation for this class was generated from the following file:

- [checkers\\_ffpack.h](#)

## 12.95 FailureFgemmCheck Class Reference

```
#include <checkers_fflas.h>
```

The documentation for this class was generated from the following file:

- [checkers\\_fflas.h](#)

## 12.96 FailureInvertCheck Class Reference

```
#include <checkers_ffpack.h>
```

The documentation for this class was generated from the following file:

- [checkers\\_ffpack.h](#)

## 12.97 FailurePLUQCheck Class Reference

```
#include <checkers_ffpack.h>
```

The documentation for this class was generated from the following file:

- [checkers\\_ffpack.h](#)

## 12.98 FailureTrsmCheck Class Reference

```
#include <checkers_fflas.h>
```

The documentation for this class was generated from the following file:

- [checkers\\_fflas.h](#)

## 12.99 FieldSimd< \_Field > Class Template Reference

### Public Types

- [using Field = \\_Field](#)
- [using Element = typename Field::Element](#)
- [using simd = Simd< typename \\_Field::Element >](#)
- [using vect\\_t = typename simd::vect\\_t](#)
- [using scalar\\_t = typename simd::scalar\\_t](#)

### Public Member Functions

- [FieldSimd \(const Field &f\)](#)
- [FieldSimd \(const Self &\)=default](#)
- [FieldSimd \(Self &&\)=default](#)
- [Self & operator= \(const Self &\)=default](#)
- [Self & operator= \(Self &&\)=default](#)
- [INLINE vect\\_t init \(vect\\_t &x, const vect\\_t a\) const](#)
- [INLINE vect\\_t init \(const vect\\_t a\) const](#)
- [INLINE vect\\_t add \(vect\\_t &c, const vect\\_t a, const vect\\_t b\)](#)
- [INLINE vect\\_t add \(const vect\\_t a, const vect\\_t b\)](#)
- [INLINE vect\\_t addin \(vect\\_t &a, const vect\\_t b\) const](#)
- [INLINE vect\\_t add\\_r \(vect\\_t &c, const vect\\_t a, const vect\\_t b\) const](#)
- [INLINE vect\\_t add\\_r \(const vect\\_t a, const vect\\_t b\) const](#)
- [INLINE vect\\_t addin\\_r \(vect\\_t &a, const vect\\_t b\) const](#)
- [INLINE vect\\_t sub \(vect\\_t &c, const vect\\_t a, const vect\\_t b\)](#)
- [INLINE vect\\_t sub \(const vect\\_t a, const vect\\_t b\)](#)
- [INLINE vect\\_t subin \(vect\\_t &a, const vect\\_t b\) const](#)
- [INLINE vect\\_t sub\\_r \(vect\\_t &c, const vect\\_t a, const vect\\_t b\) const](#)
- [INLINE vect\\_t sub\\_r \(const vect\\_t a, const vect\\_t b\) const](#)
- [INLINE vect\\_t subin\\_r \(vect\\_t &a, const vect\\_t b\) const](#)
- [INLINE vect\\_t zero \(vect\\_t &x\) const](#)



- `INLINE vect_t zero () const`
- `INLINE vect_t mod (vect_t &c) const`
- `INLINE vect_t mul (vect_t &c, const vect_t a, const vect_t b) const`
- `INLINE vect_t mul (const vect_t a, const vect_t b) const`
- `INLINE vect_t mulin (vect_t &a, const vect_t b) const`
- `INLINE vect_t mul_r (vect_t &c, const vect_t a, const vect_t b) const`
- `INLINE vect_t mul_r (const vect_t a, const vect_t b) const`
- `INLINE vect_t axpy (vect_t &r, const vect_t a, const vect_t b, const vect_t c) const`
- `INLINE vect_t axpy (const vect_t c, const vect_t a, const vect_t b) const`
- `INLINE vect_t axpyin (vect_t &c, const vect_t a, const vect_t b) const`
- `INLINE vect_t axpy_r (vect_t &r, const vect_t a, const vect_t b, const vect_t c) const`
- `INLINE vect_t axpy_r (const vect_t c, const vect_t a, const vect_t b) const`
- `INLINE vect_t axpyin_r (vect_t &c, const vect_t a, const vect_t b) const`
- `INLINE vect_t maxpy (vect_t &r, const vect_t a, const vect_t b, const vect_t c) const`
- `INLINE vect_t maxpy (const vect_t c, const vect_t a, const vect_t b) const`
- `INLINE vect_t maxpyin (vect_t &c, const vect_t a, const vect_t b) const`

### Static Public Attributes

- `static const constexpr size_t vect_size = simd::vect_size`
- `static const constexpr size_t alignment = simd::alignment`

## 12.99.1 Member Typedef Documentation

### 12.99.1.1 Field

```
template<class _Field >
using Field = _Field
```

### 12.99.1.2 Element

```
template<class _Field >
using Element = typename Field::Element
```

### 12.99.1.3 simd

```
template<class _Field >
using simd = Simd<typename _Field::Element>
```

### 12.99.1.4 vect\_t

```
template<class _Field >
using vect_t = typename simd::vect_t
```

### 12.99.1.5 scalar\_t

```
template<class _Field >
using scalar_t = typename simd::scalar_t
```

## 12.99.2 Constructor & Destructor Documentation

### 12.99.2.1 FieldSimd() [1/3]

```
template<class _Field >
FieldSimd (
 const Field & f) [inline]
```

**12.99.2.2 FieldSimd() [2/3]**

```
template<class _Field >
FieldSimd (
 const Self &) [default]
```

**12.99.2.3 FieldSimd() [3/3]**

```
template<class _Field >
FieldSimd (
 Self &&) [default]
```

**12.99.3 Member Function Documentation****12.99.3.1 operator=() [1/2]**

```
template<class _Field >
Self & operator= (
 const Self &) [default]
```

**12.99.3.2 operator=() [2/2]**

```
template<class _Field >
Self & operator= (
 Self &&) [default]
```

**12.99.3.3 init() [1/2]**

```
template<class _Field >
INLINE vect_t init (
 vect_t & x,
 const vect_t a) const [inline]
```

**12.99.3.4 init() [2/2]**

```
template<class _Field >
INLINE vect_t init (
 const vect_t a) const [inline]
```

**12.99.3.5 add() [1/2]**

```
template<class _Field >
INLINE vect_t add (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline]
```

**12.99.3.6 add() [2/2]**

```
template<class _Field >
INLINE vect_t add (
 const vect_t a,
 const vect_t b) [inline]
```

**12.99.3.7 addin()**

```
template<class _Field >
INLINE vect_t addin (
 vect_t & a,
 const vect_t b) const [inline]
```

**12.99.3.8 add\_r()** [1/2]

```
template<class _Field >
INLINE vect_t add_r (
 vect_t & c,
 const vect_t a,
 const vect_t b) const [inline]
```

**12.99.3.9 add\_r()** [2/2]

```
template<class _Field >
INLINE vect_t add_r (
 const vect_t a,
 const vect_t b) const [inline]
```

**12.99.3.10 addin\_r()**

```
template<class _Field >
INLINE vect_t addin_r (
 vect_t & a,
 const vect_t b) const [inline]
```

**12.99.3.11 sub()** [1/2]

```
template<class _Field >
INLINE vect_t sub (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline]
```

**12.99.3.12 sub()** [2/2]

```
template<class _Field >
INLINE vect_t sub (
 const vect_t a,
 const vect_t b) [inline]
```

**12.99.3.13 subin()**

```
template<class _Field >
INLINE vect_t subin (
 vect_t & a,
 const vect_t b) const [inline]
```

**12.99.3.14 sub\_r()** [1/2]

```
template<class _Field >
INLINE vect_t sub_r (
 vect_t & c,
 const vect_t a,
 const vect_t b) const [inline]
```

**12.99.3.15 sub\_r()** [2/2]

```
template<class _Field >
INLINE vect_t sub_r (
 const vect_t a,
 const vect_t b) const [inline]
```

**12.99.3.16 subin\_r()**

```
template<class _Field >
INLINE vect_t subin_r (
 vect_t & a,
 const vect_t b) const [inline]
```

**12.99.3.17 zero() [1/2]**

```
template<class _Field >
INLINE vect_t zero (
 vect_t & x) const [inline]
```

**12.99.3.18 zero() [2/2]**

```
template<class _Field >
INLINE vect_t zero () const [inline]
```

**12.99.3.19 mod()**

```
template<class _Field >
INLINE vect_t mod (
 vect_t & c) const [inline]
```

**12.99.3.20 mul() [1/2]**

```
template<class _Field >
INLINE vect_t mul (
 vect_t & c,
 const vect_t a,
 const vect_t b) const [inline]
```

**12.99.3.21 mul() [2/2]**

```
template<class _Field >
INLINE vect_t mul (
 const vect_t a,
 const vect_t b) const [inline]
```

**12.99.3.22 mulin()**

```
template<class _Field >
INLINE vect_t mulin (
 vect_t & a,
 const vect_t b) const [inline]
```

**12.99.3.23 mul\_r() [1/2]**

```
template<class _Field >
INLINE vect_t mul_r (
 vect_t & c,
 const vect_t a,
 const vect_t b) const [inline]
```

**12.99.3.24 mul\_r() [2/2]**

```
template<class _Field >
INLINE vect_t mul_r (
 const vect_t a,
 const vect_t b) const [inline]
```

**12.99.3.25 axpy()** [1/2]

```
template<class _Field >
INLINE vect_t axpy (
 vect_t & r,
 const vect_t a,
 const vect_t b,
 const vect_t c) const [inline]
```

**12.99.3.26 axpy()** [2/2]

```
template<class _Field >
INLINE vect_t axpy (
 const vect_t c,
 const vect_t a,
 const vect_t b) const [inline]
```

**12.99.3.27 axpyin()**

```
template<class _Field >
INLINE vect_t axpyin (
 vect_t & c,
 const vect_t a,
 const vect_t b) const [inline]
```

**12.99.3.28 axpy\_r()** [1/2]

```
template<class _Field >
INLINE vect_t axpy_r (
 vect_t & r,
 const vect_t a,
 const vect_t b,
 const vect_t c) const [inline]
```

**12.99.3.29 axpy\_r()** [2/2]

```
template<class _Field >
INLINE vect_t axpy_r (
 const vect_t c,
 const vect_t a,
 const vect_t b) const [inline]
```

**12.99.3.30 axpyin\_r()**

```
template<class _Field >
INLINE vect_t axpyin_r (
 vect_t & c,
 const vect_t a,
 const vect_t b) const [inline]
```

**12.99.3.31 maxpy()** [1/2]

```
template<class _Field >
INLINE vect_t maxpy (
 vect_t & r,
 const vect_t a,
 const vect_t b,
 const vect_t c) const [inline]
```

**12.99.3.32 maxpy() [2/2]**

```
template<class _Field >
INLINE vect_t maxpy (
 const vect_t c,
 const vect_t a,
 const vect_t b) const [inline]
```

**12.99.3.33 maxpyin()**

```
template<class _Field >
INLINE vect_t maxpyin (
 vect_t & c,
 const vect_t a,
 const vect_t b) const [inline]
```

**12.99.4 Field Documentation****12.99.4.1 vect\_size**

```
template<class _Field >
const constexpr size_t vect_size = simd::vect_size [static], [constexpr]
```

**12.99.4.2 alignment**

```
template<class _Field >
const constexpr size_t alignment = simd::alignment [static], [constexpr]
```

The documentation for this class was generated from the following file:

- [simd\\_modular.inl](#)

**12.100 FieldTraits< Field > Struct Template Reference**

FieldTrait.

```
#include <field-traits.h>
```

**Public Types**

- [typedef FieldCategories::GenericTag category](#)

**Static Public Attributes**

- [static const bool balanced = false](#)

**12.100.1 Detailed Description**

```
template<class Field>
struct FFLAS::FieldTraits< Field >
```

FieldTrait.

**12.100.2 Member Typedef Documentation****12.100.2.1 category**

```
template<class Field >
typedef FieldCategories::GenericTag category
```

### 12.100.3 Field Documentation

#### 12.100.3.1 balanced

```
template<class Field >
const bool balanced = false [static]
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.101 FieldTraits< FFPACK::RNSInteger< T > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef FieldCategories::UnparametricTag category](#)

### Static Public Attributes

- [static const bool balanced = false](#)

### 12.101.1 Member Typedef Documentation

#### 12.101.1.1 category

```
template<typename T >
typedef FieldCategories::UnparametricTag category
```

### 12.101.2 Field Documentation

#### 12.101.2.1 balanced

```
template<typename T >
const bool balanced = false [static]
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.102 FieldTraits< FFPACK::RNSIntegerMod< T > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef FieldCategories::ModularTag category](#)

### Static Public Attributes

- [static const bool balanced = false](#)

### 12.102.1 Member Typedef Documentation

#### 12.102.1.1 category

```
template<typename T >
typedef FieldCategories::ModularTag category
```

## 12.102.2 Field Documentation

### 12.102.2.1 balanced

```
template<typename T >
const bool balanced = false [static]
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.103 FieldTraits< Givaro::Modular< Element > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef FieldCategories::ModularTag category](#)

### Static Public Attributes

- [static const bool balanced = false](#)

## 12.103.1 Member Typedef Documentation

### 12.103.1.1 category

```
template<class Element >
typedef FieldCategories::ModularTag category
```

## 12.103.2 Field Documentation

### 12.103.2.1 balanced

```
template<class Element >
const bool balanced = false [static]
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.104 FieldTraits< Givaro::ModularBalanced< Element > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef FieldCategories::ModularTag category](#)

### Static Public Attributes

- [static const bool balanced = true](#)

## 12.104.1 Member Typedef Documentation

### 12.104.1.1 category

```
template<class Element >
typedef FieldCategories::ModularTag category
```



## 12.104.2 Field Documentation

### 12.104.2.1 balanced

```
template<class Element >
```

```
const bool balanced = true [static]
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.105 FieldTraits< Givaro::ZRing< double > > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef FieldCategories::UnparametricTag category](#)

### Static Public Attributes

- [static const bool balanced = false](#)

## 12.105.1 Member Typedef Documentation

### 12.105.1.1 category

```
typedef FieldCategories::UnparametricTag category
```

## 12.105.2 Field Documentation

### 12.105.2.1 balanced

```
const bool balanced = false [static]
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.106 FieldTraits< Givaro::ZRing< float > > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef FieldCategories::UnparametricTag category](#)

### Static Public Attributes

- [static const bool balanced = false](#)

## 12.106.1 Member Typedef Documentation

### 12.106.1.1 category

```
typedef FieldCategories::UnparametricTag category
```

## 12.106.2 Field Documentation

### 12.106.2.1 balanced

```
const bool balanced = false [static]
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.107 FieldTraits< Givaro::ZRing< Givaro::Integer > > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef FieldCategories::UnparametricTag category](#)

### Static Public Attributes

- [static const bool balanced = false](#)

## 12.107.1 Member Typedef Documentation

### 12.107.1.1 category

```
typedef FieldCategories::UnparametricTag category
```

## 12.107.2 Field Documentation

### 12.107.2.1 balanced

```
const bool balanced = false [static]
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.108 FieldTraits< Givaro::ZRing< int16\_t > > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef FieldCategories::UnparametricTag category](#)

### Static Public Attributes

- [static const bool balanced = false](#)

## 12.108.1 Member Typedef Documentation

### 12.108.1.1 category

```
typedef FieldCategories::UnparametricTag category
```

## 12.108.2 Field Documentation

### 12.108.2.1 balanced

```
const bool balanced = false [static]
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.109 FieldTraits< Givaro::ZRing< int32\_t > > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef FieldCategories::UnparametricTag category](#)

### Static Public Attributes

- [static const bool balanced = false](#)

## 12.109.1 Member Typedef Documentation

### 12.109.1.1 category

```
typedef FieldCategories::UnparametricTag category
```

## 12.109.2 Field Documentation

### 12.109.2.1 balanced

```
const bool balanced = false [static]
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.110 FieldTraits< Givaro::ZRing< int64\_t > > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef FieldCategories::UnparametricTag category](#)

### Static Public Attributes

- [static const bool balanced = false](#)

## 12.110.1 Member Typedef Documentation

### 12.110.1.1 category

```
typedef FieldCategories::UnparametricTag category
```

## 12.110.2 Field Documentation

### 12.110.2.1 balanced

```
const bool balanced = false [static]
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.111 FieldTraits< Givaro::ZRing< RecInt::ruint< K > > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef FieldCategories::UnparametricTag category](#)

### Static Public Attributes

- [static const bool balanced = false](#)

### 12.111.1 Member Typedef Documentation

#### 12.111.1.1 category

```
template<size_t K>
typedef FieldCategories::UnparametricTag category
```

### 12.111.2 Field Documentation

#### 12.111.2.1 balanced

```
template<size_t K>
const bool balanced = false [static]
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.112 FieldTraits< Givaro::ZRing< uint16\_t > > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef FieldCategories::UnparametricTag category](#)

### Static Public Attributes

- [static const bool balanced = false](#)

### 12.112.1 Member Typedef Documentation

#### 12.112.1.1 category

```
typedef FieldCategories::UnparametricTag category
```

### 12.112.2 Field Documentation

#### 12.112.2.1 balanced

```
const bool balanced = false [static]
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.113 FieldTraits< Givaro::ZRing< uint32\_t > > Struct Reference

```
#include <field-traits.h>
```

**Public Types**

- [typedef FieldCategories::UnparametricTag category](#)

**Static Public Attributes**

- [static const bool balanced = false](#)

**12.113.1 Member Typedef Documentation****12.113.1.1 category**

```
typedef FieldCategories::UnparametricTag category
```

**12.113.2 Field Documentation****12.113.2.1 balanced**

```
const bool balanced = false [static]
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

**12.114 FieldTraits< Givaro::ZRing< uint64\_t > > Struct Reference**

```
#include <field-traits.h>
```

**Public Types**

- [typedef FieldCategories::UnparametricTag category](#)

**Static Public Attributes**

- [static const bool balanced = false](#)

**12.114.1 Member Typedef Documentation****12.114.1.1 category**

```
typedef FieldCategories::UnparametricTag category
```

**12.114.2 Field Documentation****12.114.2.1 balanced**

```
const bool balanced = false [static]
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

**12.115 Fixed Struct Reference**

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

**12.116 FixedPrecIntTag Struct Reference**

Fixed precision integers above machine precision: Givaro::reclnt.

```
#include <field-traits.h>
```

### 12.116.1 Detailed Description

Fixed precision integers above machine precision: Givaro::reclnt.

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.117 ScalFunctionsBase< Element, typename enable\_if< is\_floating\_point< Element >::value >::type >::FloatingPointTestDistribution Class Reference

### Public Types

- [using IntType = typename make\\_unsigned\\_int< Element >::type](#)

### Public Member Functions

- [FloatingPointTestDistribution](#) ()
- [template<class Generator > Element operator\(\) \(Generator &g\)](#)

### 12.117.1 Member Typedef Documentation

#### 12.117.1.1 IntType

```
template<typename Element >
using IntType = typename make_unsigned_int<Element>::type
```

### 12.117.2 Constructor & Destructor Documentation

#### 12.117.2.1 FloatingPointTestDistribution()

```
template<typename Element >
FloatingPointTestDistribution () [inline]
```

### 12.117.3 Member Function Documentation

#### 12.117.3.1 operator()()

```
template<typename Element >
template<class Generator >
Element operator() (
 Generator & g) [inline]
```

The documentation for this class was generated from the following file:

- [test-simd.C](#)

## 12.118 ftrmmLeftLowerNoTransNonUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.119 ftrmmLeftLowerNoTransUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.120 ftrmmLeftLowerTransNonUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.121 ftrmmLeftLowerTransUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.122 ftrmmLeftUpperNoTransNonUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.123 ftrmmLeftUpperNoTransUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.124 ftrmmLeftUpperTransNonUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.125 ftrmmLeftUpperTransUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.126 ftrmmRightLowerNoTransNonUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.127 ftrmmRightLowerNoTransUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.128 **ftrmmRightLowerTransNonUnit**< **Element** > **Class Template Reference**

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.129 **ftrmmRightLowerTransUnit**< **Element** > **Class Template Reference**

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.130 **ftrmmRightUpperNoTransNonUnit**< **Element** > **Class Template Reference**

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.131 **ftrmmRightUpperNoTransUnit**< **Element** > **Class Template Reference**

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.132 **ftrmmRightUpperTransNonUnit**< **Element** > **Class Template Reference**

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.133 **ftrmmRightUpperTransUnit**< **Element** > **Class Template Reference**

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.134 **ftrsmLeftLowerNoTransNonUnit**< **Element** > **Class Template Reference**

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.135 **ftrsmLeftLowerNoTransUnit**< **Element** > **Class Template Reference**

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)



## 12.136 ftrsmLeftLowerTransNonUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.137 ftrsmLeftLowerTransUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.138 ftrsmLeftUpperNoTransNonUnit< Element > Class Template Reference

Computes the maximal size for delaying the modular reduction in a triangular system resolution.

### 12.138.1 Detailed Description

`template<class Element>`

`class FFLAS::Protected::ftrsmLeftUpperNoTransNonUnit< Element >`

Computes the maximal size for delaying the modular reduction in a triangular system resolution.

Compute the maximal dimension  $k$ , such that a unit diagonal triangular system of dimension  $k$  can be solved over  $\mathbb{Z}$  without overflow of the underlying floating point representation.

**Bibliography** • Dumas, Giorgi, Pernet 06, arXiv:cs/0601133.

Parameters

|     |                                      |
|-----|--------------------------------------|
| $F$ | Finite Field/Ring of the computation |
|-----|--------------------------------------|

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.139 ftrsmLeftUpperNoTransUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.140 ftrsmLeftUpperTransNonUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.141 ftrsmLeftUpperTransUnit< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.142 **ftrsmRightLowerNoTransNonUnit**< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.143 **ftrsmRightLowerNoTransUnit**< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.144 **ftrsmRightLowerTransNonUnit**< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.145 **ftrsmRightLowerTransUnit**< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.146 **ftrsmRightUpperNoTransNonUnit**< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.147 **ftrsmRightUpperNoTransUnit**< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.148 **ftrsmRightUpperTransNonUnit**< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.149 **ftrsmRightUpperTransUnit**< Element > Class Template Reference

The documentation for this class was generated from the following file:

- [fflas\\_level3.inl](#)

## 12.150 GenericTag Struct Reference

default is generic

```
#include <field-traits.h>
```

### 12.150.1 Detailed Description

default is generic

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.151 GenericTag Struct Reference

generic ring.

```
#include <field-traits.h>
```

### 12.151.1 Detailed Description

generic ring.

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.152 Grain Struct Reference

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

## 12.153 has\_minus\_eq\_impl< C > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

### Static Public Attributes

- [static constexpr bool value](#) = type::value

### 12.153.1 Field Documentation

#### 12.153.1.1 value

```
template<typename C >
```

```
constexpr bool value = type::value [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.154 has\_minus\_impl< C > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

### Static Public Attributes

- [static constexpr bool value](#) = type::value

### 12.154.1 Field Documentation

#### 12.154.1.1 value

```
template<typename C >
```

```
constexpr bool value = type::value [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.155 has\_mul\_eq\_impl< C > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

### Static Public Attributes

- [static constexpr bool value](#) = type::value

### 12.155.1 Field Documentation

#### 12.155.1.1 value

```
template<typename C >
```

```
constexpr bool value = type::value [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.156 has\_mul\_impl< C > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

### Static Public Attributes

- [static constexpr bool value](#) = type::value

### 12.156.1 Field Documentation

#### 12.156.1.1 value

```
template<typename C >
```

```
constexpr bool value = type::value [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.157 has\_operation< T > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

### Static Public Attributes

- [static constexpr bool value](#)

## 12.157.1 Field Documentation

### 12.157.1.1 value

```
template<class T >
```

```
constexpr bool value [static], [constexpr]
```

**Initial value:**

```
= (has_plus<T>::value && has_minus<T>::value && has_equal<T>::value &&
 has_plus_eq<T>::value && has_minus_eq<T>::value && has_mul<T>::value
 && has_mul_eq<T>::value)
```

The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.158 has\_plus\_eq\_impl< C > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

### Static Public Attributes

- [static constexpr bool value](#) = type::value

## 12.158.1 Field Documentation

### 12.158.1.1 value

```
template<typename C >
```

```
constexpr bool value = type::value [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.159 has\_plus\_impl< C > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

### Static Public Attributes

- [static constexpr bool value](#) = type::value

## 12.159.1 Field Documentation

### 12.159.1.1 value

```
template<typename C >
```

```
constexpr bool value = type::value [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.160 HelperFlag Struct Reference

```
#include <fflas_sparse.h>
```

### Static Public Attributes

- [static constexpr uint64\\_t none](#) = 0\_ui64
- [static constexpr uint64\\_t coo](#) = 1\_ui64
- [static constexpr uint64\\_t csr](#) = 1\_ui64 << 1
- [static constexpr uint64\\_t ell](#) = 1\_ui64 << 2
- [static constexpr uint64\\_t aut](#) = 1\_ui64 << 32
- [static constexpr uint64\\_t pm1](#) = 1\_ui64 << 33

### 12.160.1 Field Documentation

#### 12.160.1.1 none

```
constexpr uint64_t none = 0_ui64 [static], [constexpr]
```

#### 12.160.1.2 coo

```
constexpr uint64_t coo = 1_ui64 [static], [constexpr]
```

#### 12.160.1.3 csr

```
constexpr uint64_t csr = 1_ui64 << 1 [static], [constexpr]
```

#### 12.160.1.4 ell

```
constexpr uint64_t ell = 1_ui64 << 2 [static], [constexpr]
```

#### 12.160.1.5 aut

```
constexpr uint64_t aut = 1_ui64 << 32 [static], [constexpr]
```

#### 12.160.1.6 pm1

```
constexpr uint64_t pm1 = 1_ui64 << 33 [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [fflas\\_sparse.h](#)

## 12.161 HelperMod< Field, ElementTraits > Struct Template Reference

The documentation for this struct was generated from the following file:

- [fflas\\_freduce.inl](#)

## 12.162 HelperMod< Field, ElementCategories::MachineIntTag > Struct Template Reference

### Public Member Functions

- [HelperMod](#) ()
- [HelperMod](#) (const [Field](#) &F)

### Data Fields

- [Field::Element](#) p
- [double](#) invp
- [double](#) min
- [double](#) max
- [int64\\_t](#) pow50rem

### 12.162.1 Constructor & Destructor Documentation

#### 12.162.1.1 HelperMod() [1/2]

```
template<class Field >
HelperMod () [inline]
```

**12.162.1.2 HelperMod()** [2/2]

```
template<class Field >
HelperMod (
 const Field & F) [inline]
```

**12.162.2 Field Documentation****12.162.2.1 p**

```
template<class Field >
Field::Element p
```

**12.162.2.2 invp**

```
template<class Field >
double invp
```

**12.162.2.3 min**

```
template<class Field >
double min
```

**12.162.2.4 max**

```
template<class Field >
double max
```

**12.162.2.5 pow50rem**

```
template<class Field >
int64_t pow50rem
```

The documentation for this struct was generated from the following file:

- [fflas\\_freduce.inl](#)

## 12.163 HelperMod< Field, FFLAS::ElementCategories::ArbitraryPrecIntTag > Struct Template Reference

**Public Member Functions**

- [HelperMod\(\)](#)
- [HelperMod\(const Field &F\)](#)

**Data Fields**

- [Field::Element p](#)

**12.163.1 Constructor & Destructor Documentation****12.163.1.1 HelperMod()** [1/2]

```
template<class Field >
HelperMod () [inline]
```

**12.163.1.2 HelperMod() [2/2]**

```
template<class Field >
HelperMod (
 const Field & F) [inline]
```

**12.163.2 Field Documentation****12.163.2.1 p**

```
template<class Field >
Field::Element p
```

The documentation for this struct was generated from the following file:

- [fflas\\_freduce.inl](#)

## 12.164 HelperMod< Field, FFLAS::ElementCategories::FixedPrecIntTag > Struct Template Reference

**Public Member Functions**

- [HelperMod \(\)](#)
- [HelperMod \(const Field &F\)](#)

**Data Fields**

- [Field::Element p](#)

**12.164.1 Constructor & Destructor Documentation****12.164.1.1 HelperMod() [1/2]**

```
template<class Field >
HelperMod () [inline]
```

**12.164.1.2 HelperMod() [2/2]**

```
template<class Field >
HelperMod (
 const Field & F) [inline]
```

**12.164.2 Field Documentation****12.164.2.1 p**

```
template<class Field >
Field::Element p
```

The documentation for this struct was generated from the following file:

- [fflas\\_freduce.inl](#)

## 12.165 HelperMod< Field, FFLAS::ElementCategories::MachineFloatTag > Struct Template Reference

**Public Member Functions**

- [HelperMod \(\)](#)
- [HelperMod \(const Field &F\)](#)



**Data Fields**

- [Field::Element p](#)
- [Field::Element invp](#)
- [Field::Element min](#)
- [Field::Element max](#)

**12.165.1 Constructor & Destructor Documentation****12.165.1.1 HelperMod() [1/2]**

```
template<class Field >
HelperMod () [inline]
```

**12.165.1.2 HelperMod() [2/2]**

```
template<class Field >
HelperMod (
 const Field & F) [inline]
```

**12.165.2 Field Documentation****12.165.2.1 p**

```
template<class Field >
Field::Element p
```

**12.165.2.2 invp**

```
template<class Field >
Field::Element invp
```

**12.165.2.3 min**

```
template<class Field >
Field::Element min
```

**12.165.2.4 max**

```
template<class Field >
Field::Element max
```

The documentation for this struct was generated from the following file:

- [fflas\\_freduce.inl](#)

**12.166 Hybrid Struct Reference**

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

**12.167 Info Struct Reference****Public Member Functions**

- [Info \(uint64\\_t it, uint64\\_t s, uint64\\_t p\)](#)
- [Info \(\)=default](#)
- [Info \(const Info &\)=default](#)
- [Info \(Info &&\)=default](#)
- [Info & operator= \(const Info &\)=default](#)
- [Info & operator= \(Info &&\)=default](#)

**Data Fields**

- [uint64\\_t size](#) = 0
- [uint64\\_t perm](#) = 0
- [uint64\\_t begin](#) = 0

**12.167.1 Constructor & Destructor Documentation****12.167.1.1 Info() [1/4]**

```
Info (
 uint64_t it,
 uint64_t s,
 uint64_t p) [inline]
```

**12.167.1.2 Info() [2/4]**

```
Info () [default]
```

**12.167.1.3 Info() [3/4]**

```
Info (
 const Info &) [default]
```

**12.167.1.4 Info() [4/4]**

```
Info (
 Info &&) [default]
```

**12.167.2 Member Function Documentation****12.167.2.1 operator=() [1/2]**

```
Info & operator= (
 const Info &) [default]
```

**12.167.2.2 operator=() [2/2]**

```
Info & operator= (
 Info &&) [default]
```

**12.167.3 Field Documentation****12.167.3.1 size**

```
uint64_t size = 0
```

**12.167.3.2 perm**

```
uint64_t perm = 0
```

**12.167.3.3 begin**

```
uint64_t begin = 0
```

The documentation for this struct was generated from the following file:

- [csr\\_hyb\\_utils.inl](#)

## 12.168 Info Struct Reference

### Public Member Functions

- [Info](#) ([uint64\\_t](#) it, [uint64\\_t](#) s, [uint64\\_t](#) p)
- [Info](#) ()=default
- [Info](#) (const [Info](#) &)=default
- [Info](#) ([Info](#) &&)=default
- [Info](#) & [operator=](#) (const [Info](#) &)=default
- [Info](#) & [operator=](#) ([Info](#) &&)=default

### Data Fields

- [uint64\\_t](#) size = 0
- [uint64\\_t](#) perm = 0
- [uint64\\_t](#) begin = 0

### 12.168.1 Constructor & Destructor Documentation

#### 12.168.1.1 Info() [1/4]

```
Info (
 uint64_t it,
 uint64_t s,
 uint64_t p) [inline]
```

#### 12.168.1.2 Info() [2/4]

```
Info () [default]
```

#### 12.168.1.3 Info() [3/4]

```
Info (
 const Info &) [default]
```

#### 12.168.1.4 Info() [4/4]

```
Info (
 Info &&) [default]
```

### 12.168.2 Member Function Documentation

#### 12.168.2.1 operator=() [1/2]

```
Info & operator= (
 const Info &) [default]
```

#### 12.168.2.2 operator=() [2/2]

```
Info & operator= (
 Info &&) [default]
```

### 12.168.3 Field Documentation

#### 12.168.3.1 size

```
uint64_t size = 0
```

#### 12.168.3.2 perm

```
uint64_t perm = 0
```

### 12.168.3.3 begin

```
uint64_t begin = 0
```

The documentation for this struct was generated from the following file:

- [sell\\_utils.inl](#)

## 12.169 is\_all\_same< Args > Struct Template Reference

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.170 is\_all\_same< T, Args... > Struct Template Reference

### Static Public Attributes

- `static constexpr bool value` = `ALL<std::is_same<typename decay<T>::type, typename decay<Args>::type>::value...>::value`

### 12.170.1 Field Documentation

#### 12.170.1.1 value

```
template<typename T , typename... Args>
constexpr bool value = ALL<std::is_same<typename decay<T>::type, typename decay<Args>::type>::value...>::value [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.171 is\_all\_same<> Struct Reference

### Static Public Attributes

- `static constexpr bool value` = `true`

### 12.171.1 Field Documentation

#### 12.171.1.1 value

```
constexpr bool value = true [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.172 is\_simd< T > Struct Template Reference

```
#include <fflas_simd.h>
```

### Public Types

- `using type` = `std::integral_constant< bool, false >`

### Static Public Attributes

- `static const constexpr bool value` = `false`

## 12.172.1 Member Typedef Documentation

### 12.172.1.1 type

```
template<class T >
using type = std::integral_constant<bool, false>
```

## 12.172.2 Field Documentation

### 12.172.2.1 value

```
template<class T >
const constexpr bool value = false [static], [constexpr]
```

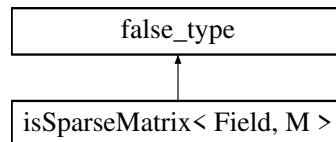
The documentation for this struct was generated from the following file:

- [fflas\\_simd.h](#)

## 12.173 isSparseMatrix< Field, M > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for isSparseMatrix< Field, M >:



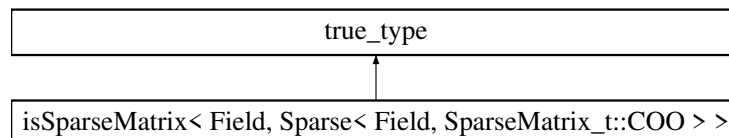
The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.174 isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::COO > > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::COO > >:



The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.175 isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::COO\_ZO > > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::COO\_ZO > >:



The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.176 isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::CSR > > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for `isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR > >`:



The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.177 isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::CSR\_HYB > > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for `isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR_HYB > >`:



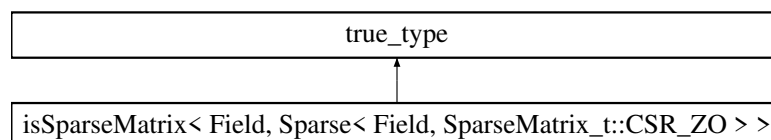
The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.178 isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::CSR\_ZO > > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for `isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::CSR_ZO > >`:



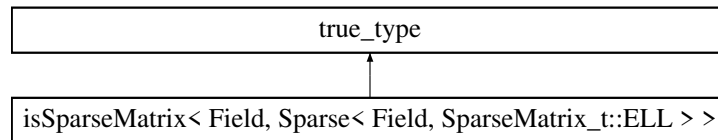
The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.179 isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::ELL > > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::ELL > >:



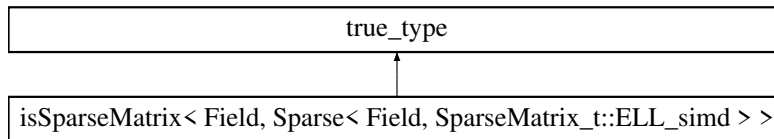
The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.180 isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::ELL\_simd > > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::ELL\_simd > >:



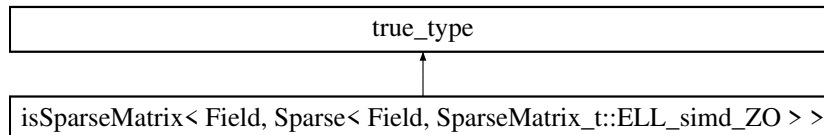
The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.181 isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::ELL\_simd\_ZO > > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::ELL\_simd\_ZO > >:



The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.182 `isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_ZO > >` Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for `isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_ZO > >`:



The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.183 `isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::HYB_ZO > >` Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for `isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::HYB_ZO > >`:



The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.184 `isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::SELL > >` Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for `isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::SELL > >`:



The documentation for this struct was generated from the following file:

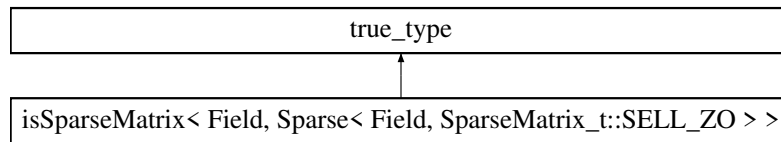
- [sparse\\_matrix\\_traits.h](#)

## 12.185 `isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::SELL_ZO > >` Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for `isSparseMatrix< Field, Sparse< Field, SparseMatrix_t::SELL_ZO > >`:





The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.186 isSparseMatrixMKLFormat< F, M > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for isSparseMatrixMKLFormat< F, M >:



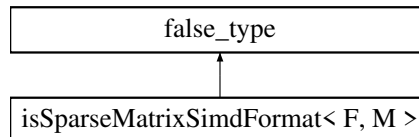
The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.187 isSparseMatrixSimdFormat< F, M > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for isSparseMatrixSimdFormat< F, M >:



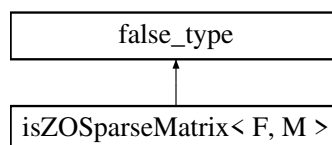
The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.188 isZOSparseMatrix< F, M > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for isZOSparseMatrix< F, M >:



The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

### 12.189 isZOSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::COO\_ZO > > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for isZOSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::COO\_ZO > >:



The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

### 12.190 isZOSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::CSR\_ZO > > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for isZOSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::CSR\_ZO > >:



The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

### 12.191 isZOSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::ELL\_simd\_ZO > > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for isZOSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::ELL\_simd\_ZO > >:



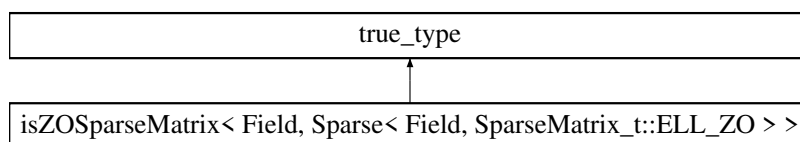
The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

### 12.192 isZOSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::ELL\_ZO > > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for isZOSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::ELL\_ZO > >:



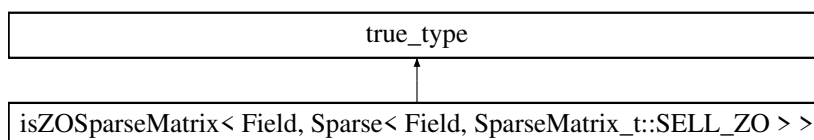
The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.193 isZOSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::SELL\_ZO > > Struct Template Reference

```
#include <sparse_matrix_traits.h>
```

Inheritance diagram for isZOSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::SELL\_ZO > >:



The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.194 Iterative Struct Reference

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

## 12.195 LazyTag Struct Reference

Performs field operations with delayed mod only when necessary. Result may not be reduced.

```
#include <field-traits.h>
```

### 12.195.1 Detailed Description

Performs field operations with delayed mod only when necessary. Result may not be reduced.

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.196 limits< T > Struct Template Reference

The documentation for this struct was generated from the following file:

- [flimits.h](#)

## 12.197 limits< char > Struct Reference

```
#include <flimits.h>
```

**Public Types**

- [typedef char T](#)

**Static Public Member Functions**

- [static constexpr char max \(\) noexcept](#)
- [static constexpr char min \(\) noexcept](#)
- [static constexpr int32\\_t digits \(\) noexcept](#)

**12.197.1 Member Typedef Documentation****12.197.1.1 T**

[typedef char T](#)

**12.197.2 Member Function Documentation****12.197.2.1 max()**

[static constexpr char max \( \) \[inline\], \[static\], \[constexpr\], \[noexcept\]](#)

**12.197.2.2 min()**

[static constexpr char min \( \) \[inline\], \[static\], \[constexpr\], \[noexcept\]](#)

**12.197.2.3 digits()**

[static constexpr int32\\_t digits \( \) \[inline\], \[static\], \[constexpr\], \[noexcept\]](#)

The documentation for this struct was generated from the following file:

- [flimits.h](#)

**12.198 limits< double > Struct Reference**

```
#include <flimits.h>
```

**Public Types**

- [typedef double T](#)

**Static Public Member Functions**

- [static constexpr int64\\_t max \(\) noexcept](#)
- [static constexpr int64\\_t min \(\) noexcept](#)
- [static constexpr int32\\_t digits \(\) noexcept](#)

**12.198.1 Member Typedef Documentation****12.198.1.1 T**

[typedef double T](#)

**12.198.2 Member Function Documentation****12.198.2.1 max()**

[static constexpr int64\\_t max \( \) \[inline\], \[static\], \[constexpr\], \[noexcept\]](#)

**12.198.2.2 min()**

```
static constexpr int64_t min () [inline], [static], [constexpr], [noexcept]
```

**12.198.2.3 digits()**

```
static constexpr int32_t digits () [inline], [static], [constexpr], [noexcept]
```

The documentation for this struct was generated from the following file:

- [flimits.h](#)

**12.199 limits< float > Struct Reference**

```
#include <flimits.h>
```

**Public Types**

- [typedef float T](#)

**Static Public Member Functions**

- [static constexpr int32\\_t max \( \) noexcept](#)
- [static constexpr int32\\_t min \( \) noexcept](#)
- [static constexpr int32\\_t digits \( \) noexcept](#)

**12.199.1 Member Typedef Documentation****12.199.1.1 T**

```
typedef float T
```

**12.199.2 Member Function Documentation****12.199.2.1 max()**

```
static constexpr int32_t max () [inline], [static], [constexpr], [noexcept]
```

**12.199.2.2 min()**

```
static constexpr int32_t min () [inline], [static], [constexpr], [noexcept]
```

**12.199.2.3 digits()**

```
static constexpr int32_t digits () [inline], [static], [constexpr], [noexcept]
```

The documentation for this struct was generated from the following file:

- [flimits.h](#)

**12.200 limits< Givaro::Integer > Struct Reference**

```
#include <flimits.h>
```

**Public Types**

- [typedef Givaro::Integer T](#)

**Static Public Member Functions**

- [static constexpr int max \( \) noexcept](#)
- [static constexpr int min \( \) noexcept](#)

## 12.200.1 Member Typedef Documentation

### 12.200.1.1 T

```
typedef Givaro::Integer T
```

## 12.200.2 Member Function Documentation

### 12.200.2.1 max()

```
static constexpr int max () [inline], [static], [constexpr], [noexcept]
```

### 12.200.2.2 min()

```
static constexpr int min () [inline], [static], [constexpr], [noexcept]
```

The documentation for this struct was generated from the following file:

- [flimits.h](#)

## 12.201 limits< int > Struct Reference

```
#include <flimits.h>
```

### Public Types

- [typedef int T](#)

### Static Public Member Functions

- [static constexpr int max \( \) noexcept](#)
- [static constexpr int min \( \) noexcept](#)
- [static constexpr int32\\_t digits \( \) noexcept](#)

## 12.201.1 Member Typedef Documentation

### 12.201.1.1 T

```
typedef int T
```

## 12.201.2 Member Function Documentation

### 12.201.2.1 max()

```
static constexpr int max () [inline], [static], [constexpr], [noexcept]
```

### 12.201.2.2 min()

```
static constexpr int min () [inline], [static], [constexpr], [noexcept]
```

### 12.201.2.3 digits()

```
static constexpr int32_t digits () [inline], [static], [constexpr], [noexcept]
```

The documentation for this struct was generated from the following file:

- [flimits.h](#)

## 12.202 limits< long > Struct Reference

```
#include <flimits.h>
```

**Public Types**

- [typedef long T](#)

**Static Public Member Functions**

- [static constexpr long max \(\) noexcept](#)
- [static constexpr long min \(\) noexcept](#)
- [static constexpr int32\\_t digits \(\) noexcept](#)

**12.202.1 Member Typedef Documentation****12.202.1.1 T**

[typedef long T](#)

**12.202.2 Member Function Documentation****12.202.2.1 max()**

[static constexpr long max \( \) \[inline\], \[static\], \[constexpr\], \[noexcept\]](#)

**12.202.2.2 min()**

[static constexpr long min \( \) \[inline\], \[static\], \[constexpr\], \[noexcept\]](#)

**12.202.2.3 digits()**

[static constexpr int32\\_t digits \( \) \[inline\], \[static\], \[constexpr\], \[noexcept\]](#)

The documentation for this struct was generated from the following file:

- [flimits.h](#)

**12.203 limits< long long > Struct Reference**

```
#include <flimits.h>
```

**Public Types**

- [typedef long long T](#)

**Static Public Member Functions**

- [static constexpr long long max \(\) noexcept](#)
- [static constexpr long long min \(\) noexcept](#)
- [static constexpr int32\\_t digits \(\) noexcept](#)

**12.203.1 Member Typedef Documentation****12.203.1.1 T**

[typedef long long T](#)

**12.203.2 Member Function Documentation****12.203.2.1 max()**

[static constexpr long long max \( \) \[inline\], \[static\], \[constexpr\], \[noexcept\]](#)

### 12.203.2.2 min()

```
static constexpr long long min () [inline], [static], [constexpr], [noexcept]
```

### 12.203.2.3 digits()

```
static constexpr int32_t digits () [inline], [static], [constexpr], [noexcept]
```

The documentation for this struct was generated from the following file:

- [flimits.h](#)

## 12.204 limits< RecInt::rint< K > > Struct Template Reference

```
#include <flimits.h>
```

### Public Types

- [typedef RecInt::ruint< K > T](#)

### Static Public Member Functions

- [static constexpr RecInt::rint< K > max \( \) noexcept](#)
- [static constexpr RecInt::rint< K > min \( \) noexcept](#)

### 12.204.1 Member Typedef Documentation

#### 12.204.1.1 T

```
template<size_t K>
typedef RecInt::ruint<K> T
```

### 12.204.2 Member Function Documentation

#### 12.204.2.1 max()

```
template<size_t K>
static constexpr RecInt::rint< K > max () [inline], [static], [constexpr], [noexcept]
```

#### 12.204.2.2 min()

```
template<size_t K>
static constexpr RecInt::rint< K > min () [inline], [static], [constexpr], [noexcept]
```

The documentation for this struct was generated from the following file:

- [flimits.h](#)

## 12.205 limits< RecInt::ruint< K > > Struct Template Reference

```
#include <flimits.h>
```

### Public Types

- [typedef RecInt::ruint< K > T](#)

### Static Public Member Functions

- [static constexpr RecInt::ruint< K > max \( \) noexcept](#)
- [static constexpr RecInt::ruint< K > min \( \) noexcept](#)



## 12.205.1 Member Typedef Documentation

### 12.205.1.1 T

```
template<size_t K>
typedef RecInt::ruint<K> T
```

## 12.205.2 Member Function Documentation

### 12.205.2.1 max()

```
template<size_t K>
static constexpr RecInt::ruint< K > max () [inline], [static], [constexpr], [noexcept]
```

### 12.205.2.2 min()

```
template<size_t K>
static constexpr RecInt::ruint< K > min () [inline], [static], [constexpr], [noexcept]
```

The documentation for this struct was generated from the following file:

- [flimits.h](#)

## 12.206 limits< short int > Struct Reference

```
#include <flimits.h>
```

### Public Types

- [typedef short int T](#)

### Static Public Member Functions

- [static constexpr short int max \( \) noexcept](#)
- [static constexpr short int min \( \) noexcept](#)
- [static constexpr int32\\_t digits \( \) noexcept](#)

## 12.206.1 Member Typedef Documentation

### 12.206.1.1 T

```
typedef short int T
```

## 12.206.2 Member Function Documentation

### 12.206.2.1 max()

```
static constexpr short int max () [inline], [static], [constexpr], [noexcept]
```

### 12.206.2.2 min()

```
static constexpr short int min () [inline], [static], [constexpr], [noexcept]
```

### 12.206.2.3 digits()

```
static constexpr int32_t digits () [inline], [static], [constexpr], [noexcept]
```

The documentation for this struct was generated from the following file:

- [flimits.h](#)

## 12.207 limits< signed char > Struct Reference

```
#include <flimits.h>
```

### Public Types

- [typedef signed char T](#)

### Static Public Member Functions

- [static constexpr signed char max \(\) noexcept](#)
- [static constexpr signed char min \(\) noexcept](#)
- [static constexpr int32\\_t digits \(\) noexcept](#)

### 12.207.1 Member Typedef Documentation

#### 12.207.1.1 T

```
typedef signed char T
```

### 12.207.2 Member Function Documentation

#### 12.207.2.1 max()

```
static constexpr signed char max () [inline], [static], [constexpr], [noexcept]
```

#### 12.207.2.2 min()

```
static constexpr signed char min () [inline], [static], [constexpr], [noexcept]
```

#### 12.207.2.3 digits()

```
static constexpr int32_t digits () [inline], [static], [constexpr], [noexcept]
```

The documentation for this struct was generated from the following file:

- [flimits.h](#)

## 12.208 limits< unsigned char > Struct Reference

```
#include <flimits.h>
```

### Public Types

- [typedef unsigned char T](#)

### Static Public Member Functions

- [static constexpr unsigned char max \(\) noexcept](#)
- [static constexpr unsigned char min \(\) noexcept](#)
- [static constexpr int32\\_t digits \(\) noexcept](#)

### 12.208.1 Member Typedef Documentation

#### 12.208.1.1 T

```
typedef unsigned char T
```

## 12.208.2 Member Function Documentation

### 12.208.2.1 max()

```
static constexpr unsigned char max () [inline], [static], [constexpr], [noexcept]
```

### 12.208.2.2 min()

```
static constexpr unsigned char min () [inline], [static], [constexpr], [noexcept]
```

### 12.208.2.3 digits()

```
static constexpr int32_t digits () [inline], [static], [constexpr], [noexcept]
```

The documentation for this struct was generated from the following file:

- [flimits.h](#)

## 12.209 limits< unsigned int > Struct Reference

```
#include <flimits.h>
```

### Public Types

- [typedef unsigned int T](#)

### Static Public Member Functions

- [static constexpr unsigned int max \( \) noexcept](#)
- [static constexpr unsigned int min \( \) noexcept](#)
- [static constexpr int32\\_t digits \( \) noexcept](#)

## 12.209.1 Member Typedef Documentation

### 12.209.1.1 T

```
typedef unsigned int T
```

## 12.209.2 Member Function Documentation

### 12.209.2.1 max()

```
static constexpr unsigned int max () [inline], [static], [constexpr], [noexcept]
```

### 12.209.2.2 min()

```
static constexpr unsigned int min () [inline], [static], [constexpr], [noexcept]
```

### 12.209.2.3 digits()

```
static constexpr int32_t digits () [inline], [static], [constexpr], [noexcept]
```

The documentation for this struct was generated from the following file:

- [flimits.h](#)

## 12.210 limits< unsigned long > Struct Reference

```
#include <flimits.h>
```

### Public Types

- [typedef unsigned long T](#)

**Static Public Member Functions**

- [static constexpr unsigned long max \(\) noexcept](#)
- [static constexpr unsigned long min \(\) noexcept](#)
- [static constexpr int32\\_t digits \(\) noexcept](#)

**12.210.1 Member Typedef Documentation****12.210.1.1 T**

```
typedef unsigned long T
```

**12.210.2 Member Function Documentation****12.210.2.1 max()**

```
static constexpr unsigned long max () [inline], [static], [constexpr], [noexcept]
```

**12.210.2.2 min()**

```
static constexpr unsigned long min () [inline], [static], [constexpr], [noexcept]
```

**12.210.2.3 digits()**

```
static constexpr int32_t digits () [inline], [static], [constexpr], [noexcept]
```

The documentation for this struct was generated from the following file:

- [flimits.h](#)

**12.211 limits< unsigned long long > Struct Reference**

```
#include <flimits.h>
```

**Public Types**

- [typedef unsigned long long T](#)

**Static Public Member Functions**

- [static constexpr unsigned long long max \(\) noexcept](#)
- [static constexpr unsigned long long min \(\) noexcept](#)
- [static constexpr int32\\_t digits \(\) noexcept](#)

**12.211.1 Member Typedef Documentation****12.211.1.1 T**

```
typedef unsigned long long T
```

**12.211.2 Member Function Documentation****12.211.2.1 max()**

```
static constexpr unsigned long long max () [inline], [static], [constexpr], [noexcept]
```

**12.211.2.2 min()**

```
static constexpr unsigned long long min () [inline], [static], [constexpr], [noexcept]
```

### 12.211.2.3 digits()

```
static constexpr int32_t digits () [inline], [static], [constexpr], [noexcept]
```

The documentation for this struct was generated from the following file:

- [flimits.h](#)

## 12.212 limits< unsigned short int > Struct Reference

```
#include <flimits.h>
```

### Public Types

- [typedef unsigned short int T](#)

### Static Public Member Functions

- [static constexpr unsigned short int max \( \) noexcept](#)
- [static constexpr unsigned short int min \( \) noexcept](#)
- [static constexpr int32\\_t digits \( \) noexcept](#)

### 12.212.1 Member Typedef Documentation

#### 12.212.1.1 T

```
typedef unsigned short int T
```

### 12.212.2 Member Function Documentation

#### 12.212.2.1 max()

```
static constexpr unsigned short int max () [inline], [static], [constexpr], [noexcept]
```

#### 12.212.2.2 min()

```
static constexpr unsigned short int min () [inline], [static], [constexpr], [noexcept]
```

#### 12.212.2.3 digits()

```
static constexpr int32_t digits () [inline], [static], [constexpr], [noexcept]
```

The documentation for this struct was generated from the following file:

- [flimits.h](#)

## 12.213 MachineFloatTag Struct Reference

float or double

```
#include <field-traits.h>
```

### 12.213.1 Detailed Description

float or double

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.214 MachineIntTag Struct Reference

short, int, long, long long, and unsigned variants

```
#include <field-traits.h>
```

### 12.214.1 Detailed Description

short, int, long, long long, and unsigned variants

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.215 MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait > Struct Template Reference

### Public Types

- [typedef MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait > Self\\_t](#)
- [typedef associatedDelayedField< constField >::type DelayedField\\_t](#)
- [typedef associatedDelayedField< constField >::field DelayedField](#)
- [typedef DelayedField::Element DFelt](#)

### Public Member Functions

- [void initC \(\)](#)
- [void initA \(\)](#)
- [void initB \(\)](#)
- [void initOut \(\)](#)
- [size\\_t MaxDelayedDim \(DFelt beta\)](#)
- [bool Aunfit \(\)](#)
- [bool Bunfit \(\)](#)
- [void setOutBounds \(const size\\_t k, const DFelt alpha, const DFelt beta\)](#)
- [bool checkA \(const Field &F, const FFLAS::FFLAS\\_TRANSPOSE ta, const size\\_t M, const size\\_t N, typename Field::ConstElement\\_ptr A, const size\\_t lda\)](#)
- [bool checkB \(const Field &F, const FFLAS::FFLAS\\_TRANSPOSE tb, const size\\_t M, const size\\_t N, typename Field::ConstElement\\_ptr B, const size\\_t ldb\)](#)
- [bool checkOut \(const Field &F, const size\\_t M, const size\\_t N, typename Field::ConstElement\\_ptr A, const size\\_t lda\)](#)
- [bool checkOut \(const Field &F, FFLAS\\_UPLO uplo, const size\\_t M, const size\\_t N, typename Field::ConstElement\\_ptr A, const size\\_t lda\)](#)
- [MMHelper \(\)](#)
- [MMHelper \(const Field &F, size\\_t m, size\\_t k, size\\_t n, ParSeqTrait \\_PS\)](#)
- [MMHelper \(const Field &F, int w, ParSeqTrait \\_PS=ParSeqTrait\(\)\)](#)
- [template<class F2, typename AlgoT2, typename FT2, typename PS2 > MMHelper \(MMHelper< F2, AlgoT2, FT2, PS2 > &WH\)](#)
- [MMHelper \(const Field &F, int w, DFelt \\_Amin, DFelt \\_Amax, DFelt \\_Bmin, DFelt \\_Bmax, DFelt \\_Cmin, DFelt \\_Cmax, ParSeqTrait \\_PS=ParSeqTrait\(\)\)](#)

### Data Fields

- [int recLevel](#)
- [DFelt FieldMin](#)
- [DFelt FieldMax](#)
- [DFelt Amin](#)
- [DFelt Amax](#)
- [DFelt Bmin](#)
- [DFelt Bmax](#)
- [DFelt Cmin](#)
- [DFelt Cmax](#)
- [DFelt Outmin](#)
- [DFelt Outmax](#)
- [DFelt MaxStorableValue](#)
- [const DelayedField\\_t delayedField](#)
- [ParSeqTrait parseq](#)

**Friends**

- std::ostream & [operator<<](#) (std::ostream &out, const Self\_t &M)

**12.215.1 Member Typedef Documentation****12.215.1.1 Self\_t**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
typedef MMHelper<Field,AlgoTrait,ModeTrait,ParSeqTrait> Self_t
```

**12.215.1.2 DelayedField\_t**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
typedef associatedDelayedField<constField>::type DelayedField_t
```

**12.215.1.3 DelayedField**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
typedef associatedDelayedField<constField>::field DelayedField
```

**12.215.1.4 DFElt**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
typedef DelayedField::Element DFElt
```

**12.215.2 Constructor & Destructor Documentation****12.215.2.1 MMHelper() [1/5]**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
MMHelper () [inline]
```

**12.215.2.2 MMHelper() [2/5]**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
MMHelper (
 const Field & F,
 size_t m,
 size_t k,
 size_t n,
 ParSeqTrait _PS) [inline]
```

**12.215.2.3 MMHelper() [3/5]**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
MMHelper (
 const Field & F,
 int w,
 ParSeqTrait _PS = ParSeqTrait()) [inline]
```

**12.215.2.4 MMHelper() [4/5]**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
template<class F2 , typename AlgoT2 , typename FT2 , typename PS2 >
MMHelper (
 MMHelper< F2, AlgoT2, FT2, PS2 > & WH) [inline]
```

**12.215.2.5 MMHelper() [5/5]**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
MMHelper (
 const Field & F,
 int w,
 DFelt _Amin,
 DFelt _Amax,
 DFelt _Bmin,
 DFelt _Bmax,
 DFelt _Cmin,
 DFelt _Cmax,
 ParSeqTrait _PS = ParSeqTrait()) [inline]
```

**12.215.3 Member Function Documentation****12.215.3.1 initC()**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
void initC () [inline]
```

**12.215.3.2 initA()**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
void initA () [inline]
```

**12.215.3.3 initB()**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
void initB () [inline]
```

**12.215.3.4 initOut()**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
void initOut () [inline]
```

**12.215.3.5 MaxDelayedDim()**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
size_t MaxDelayedDim (
 DFelt beta) [inline]
```

**12.215.3.6 Aunfit()**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
bool Aunfit () [inline]
```

**12.215.3.7 Bunfit()**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
bool Bunfit () [inline]
```

**12.215.3.8 setOutBounds()**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
void setOutBounds (
 const size_t k,
 const DFelt alpha,
 const DFelt beta) [inline]
```



**12.215.3.9 checkA()**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
bool checkA (
 const Field & F,
 const FFLAS::FFLAS_TRANSPOSE ta,
 const size_t M,
 const size_t N,
 typename Field::ConstElement_ptr A,
 const size_t lda) [inline]
```

**12.215.3.10 checkB()**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
bool checkB (
 const Field & F,
 const FFLAS::FFLAS_TRANSPOSE tb,
 const size_t M,
 const size_t N,
 typename Field::ConstElement_ptr B,
 const size_t ldb) [inline]
```

**12.215.3.11 checkOut() [1/2]**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
bool checkOut (
 const Field & F,
 const size_t M,
 const size_t N,
 typename Field::ConstElement_ptr A,
 const size_t lda) [inline]
```

**12.215.3.12 checkOut() [2/2]**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
bool checkOut (
 const Field & F,
 FFLAS_UPLO uplo,
 const size_t M,
 const size_t N,
 typename Field::ConstElement_ptr A,
 const size_t lda) [inline]
```

**12.215.4 Friends And Related Symbol Documentation****12.215.4.1 operator<<**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
std::ostream & operator<< (
 std::ostream & out,
 const Self_t & M) [friend]
```

**12.215.5 Field Documentation****12.215.5.1 recLevel**

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
int recLevel
```

#### 12.215.5.2 FieldMin

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
DFelt FieldMin
```

#### 12.215.5.3 FieldMax

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
DFelt FieldMax
```

#### 12.215.5.4 Amin

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
DFelt Amin
```

#### 12.215.5.5 Amax

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
DFelt Amax
```

#### 12.215.5.6 Bmin

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
DFelt Bmin
```

#### 12.215.5.7 Bmax

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
DFelt Bmax
```

#### 12.215.5.8 Cmin

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
DFelt Cmin
```

#### 12.215.5.9 Cmax

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
DFelt Cmax
```

#### 12.215.5.10 Outmin

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
DFelt Outmin
```

#### 12.215.5.11 Outmax

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
DFelt Outmax
```

#### 12.215.5.12 MaxStorableValue

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
DFelt MaxStorableValue
```

#### 12.215.5.13 delayedField

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
const DelayedField_t delayedField
```

#### 12.215.5.14 parseq

```
template<class Field , typename AlgoTrait , typename ModeTrait , typename ParSeqTrait >
ParSeqTrait parseq
```

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

## 12.216 MMHelper< FFPACK::RNSInteger< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait > Struct Template Reference

### Public Types

- `typedef MMHelper< FFPACK::RNSInteger< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait > Self_t`

### Public Member Functions

- `MMHelper ()`
- `MMHelper (Givaro::Integer Amax, Givaro::Integer Bmax)`
- `MMHelper (const FFPACK::RNSInteger< E > &F, size_t m, size_t n, size_t k, ParSeqTrait PS=ParSeqTrait())`
- `MMHelper (const FFPACK::RNSInteger< E > &F, int wino, ParSeqTrait PS=ParSeqTrait())`
- `template<class F2 , class A2 , class M2 , class PS2 > MMHelper (MMHelper< F2, A2, M2, PS2 > H2)`
- `void setNorm (Givaro::Integer p)`

### Data Fields

- `Givaro::Integer normA`
- `Givaro::Integer normB`
- `int recLevel`
- `ParSeqTrait parseq`

### Friends

- `std::ostream & operator<< (std::ostream &out, const Self_t &M)`

## 12.216.1 Member Typedef Documentation

### 12.216.1.1 Self\_t

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
typedef MMHelper<FFPACK::RNSInteger<E>, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait>
Self_t
```

## 12.216.2 Constructor & Destructor Documentation

### 12.216.2.1 MMHelper() [1/5]

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
MMHelper () [inline]
```

### 12.216.2.2 MMHelper() [2/5]

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
MMHelper (
 Givaro::Integer Amax,
 Givaro::Integer Bmax) [inline]
```

**12.216.2.3 MMHelper() [3/5]**

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
MMHelper (
 const FFPACK::RNSInteger< E > & F,
 size_t m,
 size_t n,
 size_t k,
 ParSeqTrait PS = ParSeqTrait()) [inline]
```

**12.216.2.4 MMHelper() [4/5]**

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
MMHelper (
 const FFPACK::RNSInteger< E > & F,
 int wino,
 ParSeqTrait PS = ParSeqTrait()) [inline]
```

**12.216.2.5 MMHelper() [5/5]**

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
template<class F2 , class A2 , class M2 , class PS2 >
MMHelper (
 MMHelper< F2, A2, M2, PS2 > H2) [inline]
```

**12.216.3 Member Function Documentation****12.216.3.1 setNorm()**

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
void setNorm (
 Givaro::Integer p) [inline]
```

**12.216.4 Friends And Related Symbol Documentation****12.216.4.1 operator<<**

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
std::ostream & operator<< (
 std::ostream & out,
 const Self_t & M) [friend]
```

**12.216.5 Field Documentation****12.216.5.1 normA**

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
Givaro::Integer normA
```

**12.216.5.2 normB**

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
Givaro::Integer normB
```

**12.216.5.3 recLevel**

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
int recLevel
```

#### 12.216.5.4 parseq

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
ParSeqTrait parseq
```

The documentation for this struct was generated from the following file:

- [fgemm\\_classical\\_mp.inl](#)

## 12.217 MMHelper< FFPACK::RNSIntegerMod< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait > Struct Template Reference

### Public Types

- `typedef MMHelper< FFPACK::RNSIntegerMod< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait > Self_t`

### Public Member Functions

- `MMHelper ()`
- `MMHelper (Givaro::Integer Amax, Givaro::Integer Bmax)`
- `MMHelper (const FFPACK::RNSIntegerMod< E > &F, size_t m, size_t n, size_t k, ParSeqTrait PS=ParSeqTrait())`
- `MMHelper (const FFPACK::RNSIntegerMod< E > &F, int wino, ParSeqTrait PS=ParSeqTrait())`
- `template<class F2, typename AlgoT2, typename FT2, typename PS2 > MMHelper (MMHelper< F2, AlgoT2, FT2, PS2 > &WH)`
- `void setNorm (Givaro::Integer p)`

### Data Fields

- `Givaro::Integer normA`
- `Givaro::Integer normB`
- `int recLevel`
- `ParSeqTrait parseq`

### Friends

- `std::ostream & operator<< (std::ostream &out, const Self_t &M)`

## 12.217.1 Member Typedef Documentation

### 12.217.1.1 Self\_t

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
typedef MMHelper<FFPACK::RNSIntegerMod<E>, AlgoTrait,ModeCategories::DefaultTag, ParSeqTrait>
Self_t
```

## 12.217.2 Constructor & Destructor Documentation

### 12.217.2.1 MMHelper() [1/5]

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
MMHelper () [inline]
```

### 12.217.2.2 MMHelper() [2/5]

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
MMHelper (
 Givaro::Integer Amax,
 Givaro::Integer Bmax) [inline]
```

**12.217.2.3 MMHelper() [3/5]**

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
MMHelper (
 const FFPACK::RNSIntegerMod< E > & F,
 size_t m,
 size_t n,
 size_t k,
 ParSeqTrait PS = ParSeqTrait()) [inline]
```

**12.217.2.4 MMHelper() [4/5]**

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
MMHelper (
 const FFPACK::RNSIntegerMod< E > & F,
 int wino,
 ParSeqTrait PS = ParSeqTrait()) [inline]
```

**12.217.2.5 MMHelper() [5/5]**

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
template<class F2 , typename AlgoT2 , typename FT2 , typename PS2 >
MMHelper (
 MMHelper< F2, AlgoT2, FT2, PS2 > & WH) [inline]
```

**12.217.3 Member Function Documentation****12.217.3.1 setNorm()**

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
void setNorm (
 Givaro::Integer p) [inline]
```

**12.217.4 Friends And Related Symbol Documentation****12.217.4.1 operator<<**

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
std::ostream & operator<< (
 std::ostream & out,
 const Self_t & M) [friend]
```

**12.217.5 Field Documentation****12.217.5.1 normA**

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
Givaro::Integer normA
```

**12.217.5.2 normB**

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
Givaro::Integer normB
```

**12.217.5.3 recLevel**

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
int recLevel
```

#### 12.217.5.4 parseq

```
template<typename E , typename AlgoTrait , typename ParSeqTrait >
ParSeqTrait parseq
```

The documentation for this struct was generated from the following file:

- [fgemm\\_classical\\_mp.inl](#)

## 12.218 MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< Dest >, ParSeqTrait > Struct Template Reference

### Public Types

- `typedef MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< Dest >, ParSeqTrait > Self_t`

### Public Member Functions

- `MMHelper ()`
- `MMHelper (const Field &F, size_t m, size_t k, size_t n, ParSeqTrait _PS)`
- `MMHelper (const Field &F, int w, ParSeqTrait _PS=ParSeqTrait())`
- `template<class F2 , typename AlgoT2 , typename FT2 , typename PS2 > MMHelper (MMHelper< F2, AlgoT2, FT2, PS2 > &WH)`

### Data Fields

- `int recLevel`
- `ParSeqTrait parseq`

### Friends

- `std::ostream & operator<< (std::ostream &out, const Self_t &M)`

## 12.218.1 Member Typedef Documentation

### 12.218.1.1 Self\_t

```
template<class Field , typename AlgoTrait , typename Dest , typename ParSeqTrait >
typedef MMHelper<Field,AlgoTrait, ModeCategories::ConvertTo<Dest>,ParSeqTrait> Self_t
```

## 12.218.2 Constructor & Destructor Documentation

### 12.218.2.1 MMHelper() [1/4]

```
template<class Field , typename AlgoTrait , typename Dest , typename ParSeqTrait >
MMHelper () [inline]
```

### 12.218.2.2 MMHelper() [2/4]

```
template<class Field , typename AlgoTrait , typename Dest , typename ParSeqTrait >
MMHelper (
 const Field & F,
 size_t m,
 size_t k,
 size_t n,
 ParSeqTrait _PS) [inline]
```

**12.218.2.3 MMHelper() [3/4]**

```
template<class Field , typename AlgoTrait , typename Dest , typename ParSeqTrait >
MMHelper (
 const Field & F,
 int w,
 ParSeqTrait _PS = ParSeqTrait()) [inline]
```

**12.218.2.4 MMHelper() [4/4]**

```
template<class Field , typename AlgoTrait , typename Dest , typename ParSeqTrait >
template<class F2 , typename AlgoT2 , typename FT2 , typename PS2 >
MMHelper (
 MMHelper< F2, AlgoT2, FT2, PS2 > & WH) [inline]
```

**12.218.3 Friends And Related Symbol Documentation****12.218.3.1 operator<<**

```
template<class Field , typename AlgoTrait , typename Dest , typename ParSeqTrait >
std::ostream & operator<< (
 std::ostream & out,
 const Self_t & M) [friend]
```

**12.218.4 Field Documentation****12.218.4.1 recLevel**

```
template<class Field , typename AlgoTrait , typename Dest , typename ParSeqTrait >
int recLevel
```

**12.218.4.2 parseq**

```
template<class Field , typename AlgoTrait , typename Dest , typename ParSeqTrait >
ParSeqTrait parseq
```

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

## 12.219 MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeqTrait > Struct Template Reference

**Public Types**

- `typedef MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeqTrait > Self_t`

**Public Member Functions**

- `MMHelper ()`
- `template<class F2 , class A2 , class M2 , class PS2 > MMHelper (MMHelper< F2, A2, M2, PS2 > H2)`
- `MMHelper (Givaro::Integer Amax, Givaro::Integer Bmax)`
- `MMHelper (const Field &F, size_t m, size_t n, size_t k, ParSeqTrait PS=ParSeqTrait())`
- `MMHelper (const Field &F, int wino, ParSeqTrait PS=ParSeqTrait())`
- `void setNorm (Givaro::Integer p)`



## Data Fields

- Givaro::Integer [normA](#)
- Givaro::Integer [normB](#)
- [int](#) [recLevel](#)
- [ParSeqTrait](#) [parseq](#)

## Friends

- [std::ostream](#) & [operator<<](#) ([std::ostream](#) &[out](#), [const Self\\_t](#) &[M](#))

## 12.219.1 Member Typedef Documentation

### 12.219.1.1 Self\_t

```
template<typename Field , typename AlgoTrait , typename ParSeqTrait >
typedef MMHelper<Field, AlgoTrait, ModeCategories::ConvertTo<ElementCategories::RNSElementTag>,
ParSeqTrait> Self_t
```

## 12.219.2 Constructor & Destructor Documentation

### 12.219.2.1 MMHelper() [1/5]

```
template<typename Field , typename AlgoTrait , typename ParSeqTrait >
MMHelper () [inline]
```

### 12.219.2.2 MMHelper() [2/5]

```
template<typename Field , typename AlgoTrait , typename ParSeqTrait >
template<class F2 , class A2 , class M2 , class PS2 >
MMHelper (
 MMHelper< F2, A2, M2, PS2 > H2) [inline]
```

### 12.219.2.3 MMHelper() [3/5]

```
template<typename Field , typename AlgoTrait , typename ParSeqTrait >
MMHelper (
 Givaro::Integer Amax,
 Givaro::Integer Bmax) [inline]
```

### 12.219.2.4 MMHelper() [4/5]

```
template<typename Field , typename AlgoTrait , typename ParSeqTrait >
MMHelper (
 const Field & F,
 size_t m,
 size_t n,
 size_t k,
 ParSeqTrait PS = ParSeqTrait()) [inline]
```

### 12.219.2.5 MMHelper() [5/5]

```
template<typename Field , typename AlgoTrait , typename ParSeqTrait >
MMHelper (
 const Field & F,
 int wino,
 ParSeqTrait PS = ParSeqTrait()) [inline]
```

### 12.219.3 Member Function Documentation

#### 12.219.3.1 setNorm()

```
template<typename Field , typename AlgoTrait , typename ParSeqTrait >
void setNorm (
 Givaro::Integer p) [inline]
```

### 12.219.4 Friends And Related Symbol Documentation

#### 12.219.4.1 operator<<

```
template<typename Field , typename AlgoTrait , typename ParSeqTrait >
std::ostream & operator<< (
 std::ostream & out,
 const Self_t & M) [friend]
```

### 12.219.5 Field Documentation

#### 12.219.5.1 normA

```
template<typename Field , typename AlgoTrait , typename ParSeqTrait >
Givaro::Integer normA
```

#### 12.219.5.2 normB

```
template<typename Field , typename AlgoTrait , typename ParSeqTrait >
Givaro::Integer normB
```

#### 12.219.5.3 recLevel

```
template<typename Field , typename AlgoTrait , typename ParSeqTrait >
int recLevel
```

#### 12.219.5.4 parseq

```
template<typename Field , typename AlgoTrait , typename ParSeqTrait >
ParSeqTrait parseq
```

The documentation for this struct was generated from the following file:

- [fgemm\\_classical\\_mp.inl](#)

## 12.220 MMHelper< Field, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait > Struct Template Reference

FGEMM Helper for Default and ConvertTo modes of operation.

#### Public Types

- `typedef MMHelper< Field, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait > Self_t`

#### Public Member Functions

- `MMHelper ()`
- `MMHelper (const Field &F, size_t m, size_t k, size_t n, ParSeqTrait _PS)`
- `MMHelper (const Field &F, int w, ParSeqTrait _PS=ParSeqTrait())`
- `template<class F2 , typename AlgoT2 , typename FT2 , typename PS2 > MMHelper (MMHelper< F2, AlgoT2, FT2, PS2 > &WH)`

## Data Fields

- [int recLevel](#)
- [ParSeqTrait parseq](#)

## Friends

- `std::ostream & operator<< (std::ostream &out, const Self_t &M)`

## 12.220.1 Detailed Description

template<[class Field](#), [typename AlgoTrait](#), [typename ParSeqTrait](#)>  
 struct FFLAS::MMHelper< [Field](#), [AlgoTrait](#), [ModeCategories::DefaultTag](#), [ParSeqTrait](#) >

FGEMM Helper for Default and ConvertTo modes of operation.

## 12.220.2 Member Typedef Documentation

### 12.220.2.1 Self\_t

```
template<class Field , typename AlgoTrait , typename ParSeqTrait >
typedef MMHelper<Field,AlgoTrait, ModeCategories::DefaultTag,ParSeqTrait> Self_t
```

## 12.220.3 Constructor & Destructor Documentation

### 12.220.3.1 MMHelper() [1/4]

```
template<class Field , typename AlgoTrait , typename ParSeqTrait >
MMHelper () [inline]
```

### 12.220.3.2 MMHelper() [2/4]

```
template<class Field , typename AlgoTrait , typename ParSeqTrait >
MMHelper (
 const Field & F,
 size_t m,
 size_t k,
 size_t n,
 ParSeqTrait _PS) [inline]
```

### 12.220.3.3 MMHelper() [3/4]

```
template<class Field , typename AlgoTrait , typename ParSeqTrait >
MMHelper (
 const Field & F,
 int w,
 ParSeqTrait _PS = ParSeqTrait()) [inline]
```

### 12.220.3.4 MMHelper() [4/4]

```
template<class Field , typename AlgoTrait , typename ParSeqTrait >
template<class F2 , typename AlgoT2 , typename FT2 , typename PS2 >
MMHelper (
 MMHelper< F2, AlgoT2, FT2, PS2 > & WH) [inline]
```

## 12.220.4 Friends And Related Symbol Documentation

### 12.220.4.1 operator<<

```
template<class Field , typename AlgoTrait , typename ParSeqTrait >
std::ostream & operator<< (
```

```
std::ostream & out,
const Self_t & M) [friend]
```

## 12.220.5 Field Documentation

### 12.220.5.1 recLevel

```
template<class Field , typename AlgoTrait , typename ParSeqTrait >
int recLevel
```

### 12.220.5.2 parseq

```
template<class Field , typename AlgoTrait , typename ParSeqTrait >
ParSeqTrait parseq
```

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

## 12.221 ModeTraits< Field > Struct Template Reference

[ModeTraits.](#)

```
#include <field-traits.h>
```

### Public Types

- [typedef ModeCategories::DefaultTag value](#)

### 12.221.1 Detailed Description

```
template<class Field>
struct FFLAS::ModeTraits< Field >
```

[ModeTraits.](#)

### 12.221.2 Member Typedef Documentation

#### 12.221.2.1 value

```
template<class Field >
typedef ModeCategories::DefaultTag value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.222 ModeTraits< Givaro::Modular< Element, Compute > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef ModeCategories::DelayedTag value](#)

### 12.222.1 Member Typedef Documentation

#### 12.222.1.1 value

```
template<typename Element , typename Compute >
typedef ModeCategories::DelayedTag value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.223 ModeTraits< Givaro::Modular< Givaro::Integer, Compute > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef ModeCategories::ConvertTo< ElementCategories::RNSElementTag > value](#)

### 12.223.1 Member Typedef Documentation

#### 12.223.1.1 value

```
template<typename Compute >
```

```
typedef ModeCategories::ConvertTo<ElementCategories::RNSElementTag> value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.224 ModeTraits< Givaro::Modular< int16\_t, Compute > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef ModeCategories::ConvertTo< ElementCategories::MachineFloatTag > value](#)

### 12.224.1 Member Typedef Documentation

#### 12.224.1.1 value

```
template<typename Compute >
```

```
typedef ModeCategories::ConvertTo<ElementCategories::MachineFloatTag> value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.225 ModeTraits< Givaro::Modular< int32\_t, Compute > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef ModeCategories::ConvertTo< ElementCategories::MachineFloatTag > value](#)

### 12.225.1 Member Typedef Documentation

#### 12.225.1.1 value

```
template<typename Compute >
```

```
typedef ModeCategories::ConvertTo<ElementCategories::MachineFloatTag> value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.226 ModeTraits< Givaro::Modular< int64\_t, uint64\_t > > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef ModeCategories::DefaultTag value](#)

### 12.226.1 Member Typedef Documentation

#### 12.226.1.1 value

```
typedef ModeCategories::DefaultTag value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.227 ModeTraits< Givaro::Modular< int8\_t, Compute > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef ModeCategories::ConvertTo< ElementCategories::MachineFloatTag > value](#)

### 12.227.1 Member Typedef Documentation

#### 12.227.1.1 value

```
template<typename Compute >
```

```
typedef ModeCategories::ConvertTo<ElementCategories::MachineFloatTag> value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.228 ModeTraits< Givaro::Modular< RecInt::ruint< K >, Compute > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef ModeCategories::ConvertTo< ElementCategories::RNSElementTag > value](#)

### 12.228.1 Member Typedef Documentation

#### 12.228.1.1 value

```
template<typename Compute , size_t K>
```

```
typedef ModeCategories::ConvertTo<ElementCategories::RNSElementTag> value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.229 ModeTraits< Givaro::Modular< uint16\_t, Compute > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef ModeCategories::ConvertTo< ElementCategories::MachineFloatTag > value](#)

### 12.229.1 Member Typedef Documentation

#### 12.229.1.1 value

```
template<typename Compute >
```

```
typedef ModeCategories::ConvertTo<ElementCategories::MachineFloatTag> value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.230 ModeTraits< Givaro::Modular< uint32\_t, Compute > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef ModeCategories::ConvertTo< ElementCategories::MachineFloatTag > value](#)

### 12.230.1 Member Typedef Documentation

#### 12.230.1.1 value

```
template<typename Compute >
```

```
typedef ModeCategories::ConvertTo<ElementCategories::MachineFloatTag> value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.231 ModeTraits< Givaro::Modular< uint8\_t, Compute > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef ModeCategories::ConvertTo< ElementCategories::MachineFloatTag > value](#)

### 12.231.1 Member Typedef Documentation

#### 12.231.1.1 value

```
template<typename Compute >
```

```
typedef ModeCategories::ConvertTo<ElementCategories::MachineFloatTag> value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.232 ModeTraits< Givaro::ModularBalanced< Element > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef ModeCategories::DelayedTag value](#)

### 12.232.1 Member Typedef Documentation

#### 12.232.1.1 value

```
template<typename Element >
typedef ModeCategories::DelayedTag value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.233 ModeTraits< Givaro::ModularBalanced< Givaro::Integer > > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef ModeCategories::ConvertTo< ElementCategories::RNSElementTag > value](#)

### 12.233.1 Member Typedef Documentation

#### 12.233.1.1 value

```
typedef ModeCategories::ConvertTo<ElementCategories::RNSElementTag> value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.234 ModeTraits< Givaro::ModularBalanced< int16\_t > > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef ModeCategories::ConvertTo< ElementCategories::MachineFloatTag > value](#)

### 12.234.1 Member Typedef Documentation

#### 12.234.1.1 value

```
typedef ModeCategories::ConvertTo<ElementCategories::MachineFloatTag> value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)



## 12.235 ModeTraits< Givaro::ModularBalanced< int32\_t > > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef ModeCategories::ConvertTo< ElementCategories::MachineFloatTag > value](#)

### 12.235.1 Member Typedef Documentation

#### 12.235.1.1 value

```
typedef ModeCategories::ConvertTo<ElementCategories::MachineFloatTag> value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.236 ModeTraits< Givaro::ModularBalanced< int8\_t > > Struct Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef ModeCategories::ConvertTo< ElementCategories::MachineFloatTag > value](#)

### 12.236.1 Member Typedef Documentation

#### 12.236.1.1 value

```
typedef ModeCategories::ConvertTo<ElementCategories::MachineFloatTag> value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.237 ModeTraits< Givaro::Montgomery< T > > Struct Template Reference

```
#include <field-traits.h>
```

### Public Types

- [typedef ModeCategories::DefaultBoundedTag value](#)

### 12.237.1 Member Typedef Documentation

#### 12.237.1.1 value

```
template<class T >
```

```
typedef ModeCategories::DefaultBoundedTag value
```

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.238 ModeTraits< Givaro::ZRing< double > > Struct Reference

```
#include <field-traits.h>
```

**Public Types**

- [typedef ModeCategories::DefaultBoundedTag value](#)

**12.238.1 Member Typedef Documentation****12.238.1.1 value**

[typedef ModeCategories::DefaultBoundedTag value](#)

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

**12.239 ModeTraits< Givaro::ZRing< float > > Struct Reference**

```
#include <field-traits.h>
```

**Public Types**

- [typedef ModeCategories::DefaultBoundedTag value](#)

**12.239.1 Member Typedef Documentation****12.239.1.1 value**

[typedef ModeCategories::DefaultBoundedTag value](#)

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

**12.240 ModeTraits< Givaro::ZRing< Givaro::Integer > > Struct Reference**

```
#include <field-traits.h>
```

**Public Types**

- [typedef ModeCategories::ConvertTo< ElementCategories::RNSElementTag > value](#)

**12.240.1 Member Typedef Documentation****12.240.1.1 value**

[typedef ModeCategories::ConvertTo<ElementCategories::RNSElementTag> value](#)

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

**12.241 ModularBalanced< T > Class Template Reference**

The documentation for this class was generated from the following file:

- [field-traits.h](#)

**12.242 ModularTag Struct Reference**

This is a modular field like e.g. `Modular<T>` or `ModularBalanced<T>`

```
#include <field-traits.h>
```

### 12.242.1 Detailed Description

This is a modular field like e.g. `Modular<T>` or `ModularBalanced<T>`  
 The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.243 Montgomery< T > Class Template Reference

The documentation for this class was generated from the following file:

- [field-traits.h](#)

## 12.244 need\_field\_characteristic< Field > Struct Template Reference

### Static Public Attributes

- [static constexpr bool value = false](#)

### 12.244.1 Field Documentation

#### 12.244.1.1 value

```
template<typename Field >
constexpr bool value = false [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [benchmark-fgemv.C](#)

## 12.245 need\_field\_characteristic< Givaro::Modular< Field > > Struct Template Reference

### Static Public Attributes

- [static constexpr bool value = true](#)

### 12.245.1 Field Documentation

#### 12.245.1.1 value

```
template<typename Field >
constexpr bool value = true [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [benchmark-fgemv.C](#)

## 12.246 need\_field\_characteristic< Givaro::ModularBalanced< Field > > Struct Template Reference

### Static Public Attributes

- [static constexpr bool value = true](#)

### 12.246.1 Field Documentation

#### 12.246.1.1 value

```
template<typename Field >
constexpr bool value = true [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [benchmark-fgemv.C](#)

## 12.247 NoSimd< T > Struct Template Reference

```
#include <fflas_simd.h>
```

### Public Types

- `using vect_t = T *`
- `using scalar_t = T`
- `using aligned_allocator = AlignedAllocator< scalar_t, Alignment(alignment)>`
- `using aligned_vector = std::vector< scalar_t, aligned_allocator >`
- `template<class Field >`  
`using is_same_element = std::is_same< typename Field::Element, T >`

### Static Public Member Functions

- `static const std::string type_string ()`
- `template<class TT >`  
`static constexpr bool valid (TT p)`
- `template<class TT >`  
`static constexpr bool compliant (TT n)`

### Static Public Attributes

- `static const constexpr size_t vect_size = 1`
- `static const constexpr size_t alignment = static_cast<size_t>(Alignment::Normal)`

## 12.247.1 Member Typedef Documentation

### 12.247.1.1 vect\_t

```
template<typename T >
using vect_t = T*
```

### 12.247.1.2 scalar\_t

```
template<typename T >
using scalar_t = T
```

### 12.247.1.3 aligned\_allocator

```
template<typename T >
using aligned_allocator = AlignedAllocator<scalar_t, Alignment(alignment)>
```

### 12.247.1.4 aligned\_vector

```
template<typename T >
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```

### 12.247.1.5 is\_same\_element

```
template<typename T >
template<class Field >
using is_same_element = std::is_same<typename Field::Element, T>
```

## 12.247.2 Member Function Documentation

### 12.247.2.1 type\_string()

```
template<typename T >
static const std::string type_string () [inline], [static]
```

### 12.247.2.2 valid()

```
template<typename T >
template<class TT >
static constexpr bool valid (
 TT p) [inline], [static], [constexpr]
```

### 12.247.2.3 compliant()

```
template<typename T >
template<class TT >
static constexpr bool compliant (
 TT n) [inline], [static], [constexpr]
```

## 12.247.3 Field Documentation

### 12.247.3.1 vect\_size

```
template<typename T >
const constexpr size_t vect_size = 1 [static], [constexpr]
```

### 12.247.3.2 alignment

```
template<typename T >
const constexpr size_t alignment = static_cast<size_t>(Alignment::Normal) [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [fflas\\_simd.h](#)

## 12.248 Parallel< C, P > Struct Template Reference

### Public Types

- [typedef C Cut](#)
- [typedef P Param](#)

### Public Member Functions

- [Parallel](#) (size\_t n=NUM\_THREADS)
- [size\\_t numthreads](#) () const
- [size\\_t & set\\_numthreads](#) (size\_t n)

### Friends

- std::ostream & [operator<<](#) (std::ostream &out, const Parallel &p)

## 12.248.1 Member Typedef Documentation

### 12.248.1.1 Cut

```
template<typename C = CuttingStrategy::Block, typename P = StrategyParameter::Threads>
typedef C Cut
```

### 12.248.1.2 Param

```
template<typename C = CuttingStrategy::Block, typename P = StrategyParameter::Threads>
typedef P Param
```

## 12.248.2 Constructor & Destructor Documentation

### 12.248.2.1 Parallel()

```
template<typename C = CuttingStrategy::Block, typename P = StrategyParameter::Threads>
Parallel (
 size_t n = NUM_THREADS) [inline]
```

## 12.248.3 Member Function Documentation

### 12.248.3.1 numthreads()

```
template<typename C = CuttingStrategy::Block, typename P = StrategyParameter::Threads>
size_t numthreads () const [inline]
```

### 12.248.3.2 set\_numthreads()

```
template<typename C = CuttingStrategy::Block, typename P = StrategyParameter::Threads>
size_t & set_numthreads (
 size_t n) [inline]
```

## 12.248.4 Friends And Related Symbol Documentation

### 12.248.4.1 operator<<

```
template<typename C = CuttingStrategy::Block, typename P = StrategyParameter::Threads>
std::ostream & operator<< (
 std::ostream & out,
 const Parallel< C, P > & p) [friend]
```

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

## 12.249 RNSInteger< RNS >::RandIter Class Reference

```
#include <rns-integer.h>
```

Inheritance diagram for RNSInteger< RNS >::RandIter:



### Public Member Functions

- [RandIter](#) (const RNSInteger< RNS > &F, uint64\_t seed=0)
- [RNS::Element & random](#) (typename RNS::Element &elt) const  
*RNS ring Element random assignement.*
- [RNS::Element random](#) () const
- [RNS::Element & operator\(\)](#) (typename RNS::Element &elt) const
- [RNS::Element operator\(\)](#) () const
- [const RNS & ring](#) () const

## 12.249.1 Constructor & Destructor Documentation

### 12.249.1.1 RandIter()

```
template<typename RNS >
RandIter (
 const RNSInteger< RNS > & F,
 uint64_t seed = 0) [inline]
```

## 12.249.2 Member Function Documentation

### 12.249.2.1 random() [1/2]

```
template<typename RNS >
RNS::Element & random (
 typename RNS::Element & elt) const [inline], [inherited]
```

RNS ring Element random assignment.

Element is supposed to be initialized

Returns

random ring Element

### 12.249.2.2 random() [2/2]

```
template<typename RNS >
RNS::Element random () const [inline], [inherited]
```

### 12.249.2.3 operator>() [1/2]

```
template<typename RNS >
RNS::Element & operator() (
 typename RNS::Element & elt) const [inline], [inherited]
```

### 12.249.2.4 operator>() [2/2]

```
template<typename RNS >
RNS::Element operator() () const [inline], [inherited]
```

### 12.249.2.5 ring()

```
template<typename RNS >
const RNS & ring () const [inline], [inherited]
```

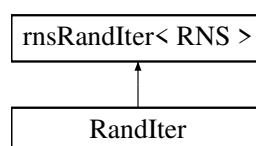
The documentation for this class was generated from the following file:

- [rns-integer.h](#)

## 12.250 RNSIntegerMod< RNS >::RandIter Class Reference

```
#include <rns-integer-mod.h>
```

Inheritance diagram for RNSIntegerMod< RNS >::RandIter:



## Public Member Functions

- [RandIter](#) ([const RNSIntegerMod](#)< [RNS](#) > &[F](#), [uint64\\_t](#) seed=0)
- [RNS::Element](#) & [random](#) ([typename RNS::Element](#) &[elt](#)) [const](#)
- [RNS::Element](#) [random](#) () [const](#)
- [RNS::Element](#) & [operator\(\)](#) ([typename RNS::Element](#) &[elt](#)) [const](#)
- [RNS::Element](#) [operator\(\)](#) () [const](#)
- [const RNS](#) & [ring](#) () [const](#)

## 12.250.1 Constructor & Destructor Documentation

### 12.250.1.1 RandIter()

```
template<typename RNS >
RandIter (
 const RNSIntegerMod< RNS > & F,
 uint64_t seed = 0) [inline]
```

## 12.250.2 Member Function Documentation

### 12.250.2.1 random() [1/2]

```
template<typename RNS >
RNS::Element & random (
 typename RNS::Element & elt) const [inline]
```

### 12.250.2.2 random() [2/2]

```
template<typename RNS >
RNS::Element random () const [inline], [inherited]
```

### 12.250.2.3 operator>() [1/2]

```
template<typename RNS >
RNS::Element & operator() (
 typename RNS::Element & elt) const [inline], [inherited]
```

### 12.250.2.4 operator>() [2/2]

```
template<typename RNS >
RNS::Element operator() () const [inline], [inherited]
```

### 12.250.2.5 ring()

```
template<typename RNS >
const RNS & ring () const [inline], [inherited]
```

The documentation for this class was generated from the following file:

- [rns-integer-mod.h](#)

## 12.251 readMyMachineType< Field, T > Struct Template Reference

```
#include <read_sparse.h>
```

## Public Types

- [typedef Field::Element](#) [Element](#)
- [typedef Field::Element\\_ptr](#) [Element\\_ptr](#)



## Public Member Functions

- [void operator\(\)](#) ([const Field &F](#), [Element &modulo](#), [Element\\_ptr val](#), [std::ifstream &file](#), [const uint64\\_t dims](#), [const mask\\_t data\\_type](#), [const mask\\_t field\\_desc](#))

## 12.251.1 Member Typedef Documentation

### 12.251.1.1 Element

```
template<class Field , class T >
typedef Field::Element Element
```

### 12.251.1.2 Element\_ptr

```
template<class Field , class T >
typedef Field::Element_ptr Element_ptr
```

## 12.251.2 Member Function Documentation

### 12.251.2.1 operator>()

```
template<class Field , typename T >
void operator() (
 const Field & F,
 Element & modulo,
 Element_ptr val,
 std::ifstream & file,
 const uint64_t dims,
 const mask_t data_type,
 const mask_t field_desc)
```

The documentation for this struct was generated from the following file:

- [read\\_sparse.h](#)

## 12.252 readMyMachineType< Field, mpz\_t > Struct Template Reference

```
#include <read_sparse.h>
```

## Public Types

- [typedef Field::Element Element](#)
- [typedef Field::Element\\_ptr Element\\_ptr](#)

## Public Member Functions

- [void operator\(\)](#) ([const Field &F](#), [Element &modulo](#), [Element\\_ptr val](#), [std::ifstream &file](#), [const uint64\\_t dims](#), [const mask\\_t data\\_type](#), [const mask\\_t field\\_desc](#))

## 12.252.1 Member Typedef Documentation

### 12.252.1.1 Element

```
template<class Field >
typedef Field::Element Element
```

### 12.252.1.2 Element\_ptr

```
template<class Field >
typedef Field::Element_ptr Element_ptr
```

## 12.252.2 Member Function Documentation

### 12.252.2.1 operator>()

```
template<class Field >
void operator() (
 const Field & F,
 typename Field::Element & modulo,
 typename Field::Element_ptr val,
 std::ifstream & file,
 const uint64_t dims,
 const mask_t data_type,
 const mask_t field_desc)
```

The documentation for this struct was generated from the following file:

- [read\\_sparse.h](#)

## 12.253 Recursive Struct Reference

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

## 12.254 Recursive Struct Reference

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

## 12.255 rint< K > Class Template Reference

The documentation for this class was generated from the following file:

- [field-traits.h](#)

## 12.256 rns\_double Struct Reference

```
#include <rns-double.h>
```

### Public Types

- `typedef Givaro::Integer integer`
- `typedef Givaro::Modular< double > ModField`
- `typedef double BasisElement`
- `typedef rns_double_elt Element`
- `typedef rns_double_elt_ptr Element_ptr`
- `typedef rns_double_elt_cstptr ConstElement_ptr`

### Public Member Functions

- `rns_double (const integer &bound, size_t pbits, bool rnsmod=false, long seed=time(NULL))`
- `rns_double (size_t pbits, size_t size, long seed=time(NULL))`
- `template<typename Vect >`  
`rns_double (const Vect &basis, bool rnsmod=false, long seed=time(NULL))`
- `rns_double (const RNSIntegerMod< rns_double > &basis, bool rnsmod=false, long seed=time(NULL))`
- `void precompute_cst (size_t K=0)`

- `template<typename T>`  
`void init (size_t m, size_t n, double *Arns, size_t rda, const T *A, size_t lda, const integer &maxA, bool RNS_MAJOR=false) const`
- `void init (size_t m, size_t n, double *Arns, size_t rda, const integer *A, size_t lda, size_t k, bool RNS_MAJOR=false) const`
- `void init_transpose (size_t m, size_t n, double *Arns, size_t rda, const integer *A, size_t lda, size_t k, bool RNS_MAJOR=false) const`
- `void convert (size_t m, size_t n, integer gamma, integer *A, size_t lda, const double *Arns, size_t rda, bool RNS_MAJOR=false) const`
- `void convert_transpose (size_t m, size_t n, integer gamma, integer *A, size_t lda, const double *Arns, size_t rda, bool RNS_MAJOR=false) const`
- `void reduce (size_t n, double *Arns, size_t rda, bool RNS_MAJOR=false) const`
- `template<size_t K>`  
`void init (size_t m, size_t n, double *Arns, size_t rda, const Reclnt::ruint< K > *A, size_t lda, size_t k, bool RNS_MAJOR=false) const`
- `template<size_t K>`  
`void convert (size_t m, size_t n, integer gamma, Reclnt::ruint< K > *A, size_t lda, const double *Arns, size_t rda, integer p=0, bool RNS_MAJOR=false) const`

## Data Fields

- `std::vector< double, AlignedAllocator< double, Alignment::CACHE_LINE > > _basis`
- `std::vector< double, AlignedAllocator< double, Alignment::CACHE_LINE > > _basisMax`
- `std::vector< double, AlignedAllocator< double, Alignment::CACHE_LINE > > _negbasis`
- `std::vector< double, AlignedAllocator< double, Alignment::CACHE_LINE > > _invbasis`
- `std::vector< ModField > _field_rns`
- `integer _M`
- `std::vector< integer > _Mi`
- `std::vector< double > _MMi`
- `std::vector< double > _crt_in`
- `std::vector< double > _crt_out`
- `size_t _size`
- `size_t _pbits`
- `size_t _ldm`
- `integer _mi_sum`

## 12.256.1 Member Typedef Documentation

### 12.256.1.1 integer

```
typedef Givaro::Integer integer
```

### 12.256.1.2 ModField

```
typedef Givaro::Modular<double> ModField
```

### 12.256.1.3 BasisElement

```
typedef double BasisElement
```

### 12.256.1.4 Element

```
typedef rns_double_elt Element
```

### 12.256.1.5 Element\_ptr

```
typedef rns_double_elt_ptr Element_ptr
```

### 12.256.1.6 ConstElement\_ptr

```
typedef rns_double_elt_cstptr ConstElement_ptr
```

## 12.256.2 Constructor & Destructor Documentation

### 12.256.2.1 rns\_double() [1/4]

```
rns_double (
 const integer & bound,
 size_t pbits,
 bool rnsmod = false,
 long seed = time(NULL)) [inline]
```

### 12.256.2.2 rns\_double() [2/4]

```
rns_double (
 size_t pbits,
 size_t size,
 long seed = time(NULL)) [inline]
```

### 12.256.2.3 rns\_double() [3/4]

```
template<typename Vect >
rns_double (
 const Vect & basis,
 bool rnsmod = false,
 long seed = time(NULL)) [inline]
```

### 12.256.2.4 rns\_double() [4/4]

```
rns_double (
 const RNSIntegerMod< rns_double > & basis,
 bool rnsmod = false,
 long seed = time(NULL)) [inline]
```

## 12.256.3 Member Function Documentation

### 12.256.3.1 precompute\_cst()

```
void precompute_cst (
 size_t K = 0) [inline]
```

### 12.256.3.2 init() [1/3]

```
template<typename T >
void init (
 size_t m,
 size_t n,
 double * Arns,
 size_t rda,
 const T * A,
 size_t lda,
 const integer & maxA,
 bool RNS_MAJOR = false) const [inline]
```

### 12.256.3.3 init() [2/3]

```
void init (
 size_t m,
```

```

 size_t n,
 double * Arns,
 size_t rda,
 const integer * A,
 size_t lda,
 size_t k,
 bool RNS_MAJOR = false) const [inline]

```

#### 12.256.3.4 init\_transpose()

```

void init_transpose (
 size_t m,
 size_t n,
 double * Arns,
 size_t rda,
 const integer * A,
 size_t lda,
 size_t k,
 bool RNS_MAJOR = false) const [inline]

```

#### 12.256.3.5 convert() [1/2]

```

void convert (
 size_t m,
 size_t n,
 integer gamma,
 integer * A,
 size_t lda,
 const double * Arns,
 size_t rda,
 bool RNS_MAJOR = false) const [inline]

```

#### 12.256.3.6 convert\_transpose()

```

void convert_transpose (
 size_t m,
 size_t n,
 integer gamma,
 integer * A,
 size_t lda,
 const double * Arns,
 size_t rda,
 bool RNS_MAJOR = false) const [inline]

```

#### 12.256.3.7 reduce()

```

void reduce (
 size_t n,
 double * Arns,
 size_t rda,
 bool RNS_MAJOR = false) const [inline]

```

#### 12.256.3.8 init() [3/3]

```

template<size_t K>
void init (
 size_t m,
 size_t n,
 double * Arns,

```

```

size_t rda,
const RecInt::ruint< K > * A,
size_t lda,
size_t k,
bool RNS_MAJOR = false) const [inline]

```

### 12.256.3.9 convert() [2/2]

```

template<size_t K>
void convert (
 size_t m,
 size_t n,
 integer gamma,
 RecInt::ruint< K > * A,
 size_t lda,
 const double * Arns,
 size_t rda,
 integer p = 0,
 bool RNS_MAJOR = false) const [inline]

```

## 12.256.4 Field Documentation

### 12.256.4.1 \_basis

```
std::vector<double, AlignedAllocator<double, Alignment::CACHE_LINE> > _basis
```

### 12.256.4.2 \_basisMax

```
std::vector<double, AlignedAllocator<double, Alignment::CACHE_LINE> > _basisMax
```

### 12.256.4.3 \_negbasis

```
std::vector<double, AlignedAllocator<double, Alignment::CACHE_LINE> > _negbasis
```

### 12.256.4.4 \_invbasis

```
std::vector<double, AlignedAllocator<double, Alignment::CACHE_LINE> > _invbasis
```

### 12.256.4.5 \_field\_rns

```
std::vector<ModField> _field_rns
```

### 12.256.4.6 \_M

```
integer _M
```

### 12.256.4.7 \_Mi

```
std::vector<integer> _Mi
```

### 12.256.4.8 \_MMi

```
std::vector<double> _MMi
```

### 12.256.4.9 \_crt\_in

```
std::vector<double> _crt_in
```

### 12.256.4.10 \_crt\_out

```
std::vector<double> _crt_out
```

**12.256.4.11** `_size``size_t` `_size`**12.256.4.12** `_pbits``size_t` `_pbits`**12.256.4.13** `_ldm``size_t` `_ldm`**12.256.4.14** `_mi_sum``integer` `_mi_sum`

The documentation for this struct was generated from the following files:

- [rns-double.h](#)
- [rns-double-recint.inl](#)
- [rns-double.inl](#)

**12.257** `rns_double_elt` Struct Reference

```
#include <rns-double-elt.h>
```

Inheritance diagram for `rns_double_elt`:

**Public Member Functions**

- [rns\\_double\\_elt](#) ()
- [~rns\\_double\\_elt](#) ()
- [rns\\_double\\_elt](#) ([double](#) \*[p](#), [size\\_t](#) [r](#), [size\\_t](#) [a](#)=false)
- [rns\\_double\\_elt\\_ptr](#) operator& ()
- [rns\\_double\\_elt\\_cstptr](#) operator& () const
- [rns\\_double\\_elt](#) (const [rns\\_double\\_elt](#) &[x](#))

**Data Fields**

- [double](#) \* [\\_ptr](#)
- [size\\_t](#) [\\_stride](#)
- [bool](#) [\\_alloc](#)

**12.257.1** Constructor & Destructor Documentation**12.257.1.1** `rns_double_elt()` [1/3]

```
rns_double_elt () [inline]
```

**12.257.1.2** `~rns_double_elt()`

```
~rns_double_elt () [inline]
```

**12.257.1.3 rns\_double\_elt()** [2/3]

```
rns_double_elt (
 double * p,
 size_t r,
 size_t a = false) [inline]
```

**12.257.1.4 rns\_double\_elt()** [3/3]

```
rns_double_elt (
 const rns_double_elt & x) [inline]
```

**12.257.2 Member Function Documentation****12.257.2.1 operator&()** [1/2]

```
rns_double_elt_ptr operator& () [inline]
```

**12.257.2.2 operator&()** [2/2]

```
rns_double_elt_cstptr operator& () const [inline]
```

**12.257.3 Field Documentation****12.257.3.1 \_ptr**

```
double* _ptr
```

**12.257.3.2 \_stride**

```
size_t _stride
```

**12.257.3.3 \_alloc**

```
bool _alloc
```

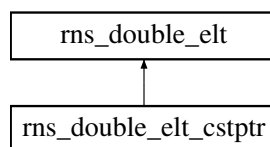
The documentation for this struct was generated from the following file:

- [rns-double-elt.h](#)

**12.258 rns\_double\_elt\_cstptr Struct Reference**

```
#include <rns-double-elt.h>
```

Inheritance diagram for rns\_double\_elt\_cstptr:

**Public Member Functions**

- [rns\\_double\\_elt\\_cstptr](#) ()
- [rns\\_double\\_elt\\_cstptr](#) (double \*p, size\_t r)
- [rns\\_double\\_elt\\_cstptr](#) (const rns\_double\_elt\_ptr &x)
- [rns\\_double\\_elt\\_cstptr](#) (const rns\_double\_elt\_cstptr &x)
- [rns\\_double\\_elt\\_cstptr](#) (rns\_double\_elt\_cstptr &&)=default
- [rns\\_double\\_elt\\_cstptr \\* operator&](#) ()
- [rns\\_double\\_elt & operator\\*](#) () const



- `rns_double_elt operator[] (size_t i) const`
- `rns_double_elt & operator[] (size_t i)`
- `rns_double_elt_cstptr operator++ ()`
- `rns_double_elt_cstptr operator-- ()`
- `rns_double_elt_cstptr operator+ (size_t inc) const`
- `rns_double_elt_cstptr operator- (size_t inc) const`
- `rns_double_elt_cstptr & operator+= (size_t inc)`
- `rns_double_elt_cstptr & operator-= (size_t inc)`
- `rns_double_elt_cstptr & operator= (const rns_double_elt_cstptr &x)`
- `bool operator< (const rns_double_elt_cstptr &x)`
- `bool operator!= (const rns_double_elt_cstptr &x)`
- `rns_double_elt_cstptr operator& () const`

### Data Fields

- `rns_double_elt other`
- `double * _ptr`
- `size_t _stride`
- `bool _alloc`

## 12.258.1 Constructor & Destructor Documentation

### 12.258.1.1 `rns_double_elt_cstptr()` [1/5]

```
rns_double_elt_cstptr () [inline]
```

### 12.258.1.2 `rns_double_elt_cstptr()` [2/5]

```
rns_double_elt_cstptr (
 double * p,
 size_t r) [inline]
```

### 12.258.1.3 `rns_double_elt_cstptr()` [3/5]

```
rns_double_elt_cstptr (
 const rns_double_elt_ptr & x) [inline]
```

### 12.258.1.4 `rns_double_elt_cstptr()` [4/5]

```
rns_double_elt_cstptr (
 const rns_double_elt_cstptr & x) [inline]
```

### 12.258.1.5 `rns_double_elt_cstptr()` [5/5]

```
rns_double_elt_cstptr (
 rns_double_elt_cstptr &&) [default]
```

## 12.258.2 Member Function Documentation

### 12.258.2.1 `operator&()` [1/2]

```
rns_double_elt_cstptr * operator& () [inline]
```

### 12.258.2.2 `operator*()`

```
rns_double_elt & operator* () const [inline]
```

**12.258.2.3 operator[]() [1/2]**

```
rns_double_elt operator[] (
 size_t i) const [inline]
```

**12.258.2.4 operator[]() [2/2]**

```
rns_double_elt & operator[] (
 size_t i) [inline]
```

**12.258.2.5 operator++()**

```
rns_double_elt_cstptr operator++ () [inline]
```

**12.258.2.6 operator--()**

```
rns_double_elt_cstptr operator-- () [inline]
```

**12.258.2.7 operator+()**

```
rns_double_elt_cstptr operator+ (
 size_t inc) const [inline]
```

**12.258.2.8 operator-()**

```
rns_double_elt_cstptr operator- (
 size_t inc) const [inline]
```

**12.258.2.9 operator+=()**

```
rns_double_elt_cstptr & operator+= (
 size_t inc) [inline]
```

**12.258.2.10 operator-=()**

```
rns_double_elt_cstptr & operator-= (
 size_t inc) [inline]
```

**12.258.2.11 operator=()**

```
rns_double_elt_cstptr & operator= (
 const rns_double_elt_cstptr & x) [inline]
```

**12.258.2.12 operator<()**

```
bool operator< (
 const rns_double_elt_cstptr & x) [inline]
```

**12.258.2.13 operator"!=()**

```
bool operator!= (
 const rns_double_elt_cstptr & x) [inline]
```

**12.258.2.14 operator&() [2/2]**

```
rns_double_elt_cstptr operator& () const [inline], [inherited]
```

## 12.258.3 Field Documentation

### 12.258.3.1 other

`rns_double_elt` other

### 12.258.3.2 \_ptr

`double*` \_ptr [inherited]

### 12.258.3.3 \_stride

`size_t` \_stride [inherited]

### 12.258.3.4 \_alloc

`bool` \_alloc [inherited]

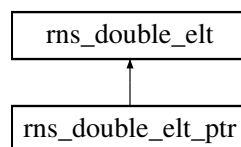
The documentation for this struct was generated from the following file:

- [rns-double-elt.h](#)

## 12.259 rns\_double\_elt\_ptr Struct Reference

```
#include <rns-double-elt.h>
```

Inheritance diagram for `rns_double_elt_ptr`:



### Public Member Functions

- [rns\\_double\\_elt\\_ptr](#) ()
- [rns\\_double\\_elt\\_ptr](#) (`double *p, size_t r`)
- [rns\\_double\\_elt\\_ptr](#) (`const rns_double_elt_ptr &x`)
- [rns\\_double\\_elt\\_ptr](#) (`const rns_double_elt_cstptr &x`)
- [rns\\_double\\_elt\\_ptr](#) (`rns_double_elt_ptr &&`)=`default`
- [rns\\_double\\_elt\\_ptr \\* operator&](#) ()
- [rns\\_double\\_elt & operator\\*](#) ()
- [rns\\_double\\_elt operator\[\]](#) (`size_t i`) `const`
- [rns\\_double\\_elt & operator\[\]](#) (`size_t i`)
- [rns\\_double\\_elt\\_ptr operator++](#) ()
- [rns\\_double\\_elt\\_ptr operator--](#) ()
- [rns\\_double\\_elt\\_ptr operator+](#) (`size_t inc`)
- [rns\\_double\\_elt\\_ptr operator-](#) (`size_t inc`)
- [rns\\_double\\_elt\\_ptr & operator+=](#) (`size_t inc`)
- [rns\\_double\\_elt\\_ptr & operator-=](#) (`size_t inc`)
- [rns\\_double\\_elt\\_ptr & operator=](#) (`const rns_double_elt_ptr &x`)
- [bool operator<](#) (`const rns_double_elt_ptr &x`)
- [bool operator!=](#) (`const rns_double_elt_ptr &x`)
- [rns\\_double\\_elt\\_cstptr operator&](#) () `const`

**Data Fields**

- [rns\\_double\\_elt](#) other
- [double \\* \\_ptr](#)
- [size\\_t stride](#)
- [bool \\_alloc](#)

**12.259.1 Constructor & Destructor Documentation****12.259.1.1 [rns\\_double\\_elt\\_ptr\(\)](#) [1/5]**

```
rns_double_elt_ptr () [inline]
```

**12.259.1.2 [rns\\_double\\_elt\\_ptr\(\)](#) [2/5]**

```
rns_double_elt_ptr (
 double * p,
 size_t r) [inline]
```

**12.259.1.3 [rns\\_double\\_elt\\_ptr\(\)](#) [3/5]**

```
rns_double_elt_ptr (
 const rns_double_elt_ptr & x) [inline]
```

**12.259.1.4 [rns\\_double\\_elt\\_ptr\(\)](#) [4/5]**

```
rns_double_elt_ptr (
 const rns_double_elt_cstptr & x) [inline]
```

**12.259.1.5 [rns\\_double\\_elt\\_ptr\(\)](#) [5/5]**

```
rns_double_elt_ptr (
 rns_double_elt_ptr &&) [default]
```

**12.259.2 Member Function Documentation****12.259.2.1 [operator&\(\)](#) [1/2]**

```
rns_double_elt_ptr * operator& () [inline]
```

**12.259.2.2 [operator\\*\(\)](#)**

```
rns_double_elt & operator* () [inline]
```

**12.259.2.3 [operator\[\]\(\)](#) [1/2]**

```
rns_double_elt operator[] (
 size_t i) const [inline]
```

**12.259.2.4 [operator\[\]\(\)](#) [2/2]**

```
rns_double_elt & operator[] (
 size_t i) [inline]
```

**12.259.2.5 [operator++\(\)](#)**

```
rns_double_elt_ptr operator++ () [inline]
```

**12.259.2.6 [operator--\(\)](#)**

```
rns_double_elt_ptr operator-- () [inline]
```

**12.259.2.7 operator+()**

```
rns_double_elt_ptr operator+ (
 size_t inc) [inline]
```

**12.259.2.8 operator-()**

```
rns_double_elt_ptr operator- (
 size_t inc) [inline]
```

**12.259.2.9 operator+=()**

```
rns_double_elt_ptr & operator+= (
 size_t inc) [inline]
```

**12.259.2.10 operator-=()**

```
rns_double_elt_ptr & operator-= (
 size_t inc) [inline]
```

**12.259.2.11 operator=()**

```
rns_double_elt_ptr & operator= (
 const rns_double_elt_ptr & x) [inline]
```

**12.259.2.12 operator<()**

```
bool operator< (
 const rns_double_elt_ptr & x) [inline]
```

**12.259.2.13 operator"!="()**

```
bool operator!= (
 const rns_double_elt_ptr & x) [inline]
```

**12.259.2.14 operator&() [2/2]**

```
rns_double_elt_cstptr operator& () const [inline], [inherited]
```

**12.259.3 Field Documentation****12.259.3.1 other**

```
rns_double_elt other
```

**12.259.3.2 \_ptr**

```
double* _ptr [inherited]
```

**12.259.3.3 \_stride**

```
size_t _stride [inherited]
```

**12.259.3.4 \_alloc**

```
bool _alloc [inherited]
```

The documentation for this struct was generated from the following file:

- [rns-double-elt.h](#)

## 12.260 rns\_double\_extended Struct Reference

```
#include <rns-double.h>
```

### Public Types

- `typedef Givaro::Integer integer`
- `typedef Givaro::ModularExtended< double > ModField`
- `typedef double BasisElement`
- `typedef rns_double_elt Element`
- `typedef rns_double_elt_ptr Element_ptr`
- `typedef rns_double_elt_cstptr ConstElement_ptr`

### Public Member Functions

- `rns_double_extended (const integer &bound, size_t pbits, bool rnsmod=false, long seed=time(NULL))`
- `rns_double_extended (size_t pbits, size_t size, long seed=time(NULL))`
- `template<typename Vect >  
rns_double_extended (const Vect &basis, bool rnsmod=false, long seed=time(NULL))`
- `void precompute_cst ()`
- `void init (size_t m, size_t n, double *Arns, size_t rda, const integer *A, size_t lda, const integer &maxA, bool RNS_MAJOR=false) const`
- `void init (size_t m, size_t n, double *Arns, size_t rda, const integer *A, size_t lda, size_t k, bool RNS_MAJOR=false)`
- `void convert (size_t m, size_t n, integer gamma, integer *A, size_t lda, const double *Arns, size_t rda, bool RNS_MAJOR=false)`
- `void init (size_t m, double *Arns, const integer *A, size_t lda) const`
- `void convert (size_t m, integer *A, const double *Arns) const`
- `void reduce (size_t n, double *Arns, size_t rda, bool RNS_MAJOR=false) const`

### Data Fields

- `std::vector< double, AlignedAllocator< double, Alignment::CACHE_LINE > > _basis`
- `std::vector< double, AlignedAllocator< double, Alignment::CACHE_LINE > > _basisMax`
- `std::vector< double, AlignedAllocator< double, Alignment::CACHE_LINE > > _negbasis`
- `std::vector< double, AlignedAllocator< double, Alignment::CACHE_LINE > > _invbasis`
- `std::vector< ModField > _field_rns`
- `integer _M`
- `std::vector< integer > _Mi`
- `std::vector< double > _MMi`
- `std::vector< double > _crt_in`
- `std::vector< double > _crt_out`
- `size_t _size`
- `size_t _pbits`
- `size_t _ldm`

### 12.260.1 Member Typedef Documentation

#### 12.260.1.1 integer

```
typedef Givaro::Integer integer
```

#### 12.260.1.2 ModField

```
typedef Givaro::ModularExtended<double> ModField
```

### 12.260.1.3 BasisElement

```
typedef double BasisElement
```

### 12.260.1.4 Element

```
typedef rns_double_elt Element
```

### 12.260.1.5 Element\_ptr

```
typedef rns_double_elt_ptr Element_ptr
```

### 12.260.1.6 ConstElement\_ptr

```
typedef rns_double_elt_cstptr ConstElement_ptr
```

## 12.260.2 Constructor & Destructor Documentation

### 12.260.2.1 rns\_double\_extended() [1/3]

```
rns_double_extended (
 const integer & bound,
 size_t pbits,
 bool rnsmod = false,
 long seed = time(NULL)) [inline]
```

### 12.260.2.2 rns\_double\_extended() [2/3]

```
rns_double_extended (
 size_t pbits,
 size_t size,
 long seed = time(NULL)) [inline]
```

### 12.260.2.3 rns\_double\_extended() [3/3]

```
template<typename Vect >
rns_double_extended (
 const Vect & basis,
 bool rnsmod = false,
 long seed = time(NULL)) [inline]
```

## 12.260.3 Member Function Documentation

### 12.260.3.1 precompute\_cst()

```
void precompute_cst () [inline]
```

### 12.260.3.2 init() [1/3]

```
void init (
 size_t m,
 size_t n,
 double * Arns,
 size_t rda,
 const integer * A,
 size_t lda,
 const integer & maxA,
 bool RNS_MAJOR = false) const [inline]
```

**12.260.3.3 init() [2/3]**

```

void init (
 size_t m,
 size_t n,
 double * Arns,
 size_t rda,
 const integer * A,
 size_t lda,
 size_t k,
 bool RNS_MAJOR = false) [inline]

```

**12.260.3.4 convert() [1/2]**

```

void convert (
 size_t m,
 size_t n,
 integer gamma,
 integer * A,
 size_t lda,
 const double * Arns,
 size_t rda,
 bool RNS_MAJOR = false) [inline]

```

**12.260.3.5 init() [3/3]**

```

void init (
 size_t m,
 double * Arns,
 const integer * A,
 size_t lda) const [inline]

```

**12.260.3.6 convert() [2/2]**

```

void convert (
 size_t m,
 integer * A,
 const double * Arns) const [inline]

```

**12.260.3.7 reduce()**

```

void reduce (
 size_t n,
 double * Arns,
 size_t rda,
 bool RNS_MAJOR = false) const [inline]

```

**12.260.4 Field Documentation****12.260.4.1 \_basis**

```
std::vector<double, AlignedAllocator<double, Alignment::CACHE_LINE> > _basis
```

**12.260.4.2 \_basisMax**

```
std::vector<double, AlignedAllocator<double, Alignment::CACHE_LINE> > _basisMax
```

**12.260.4.3 \_negbasis**

```
std::vector<double, AlignedAllocator<double, Alignment::CACHE_LINE> > _negbasis
```



**12.260.4.4** `_invbasis`

```
std::vector<double, AlignedAllocator<double, Alignment::CACHE_LINE> > _invbasis
```

**12.260.4.5** `_field_rns`

```
std::vector<ModField> _field_rns
```

**12.260.4.6** `_M`

```
integer _M
```

**12.260.4.7** `_Mi`

```
std::vector<integer> _Mi
```

**12.260.4.8** `_MMi`

```
std::vector<double> _MMi
```

**12.260.4.9** `_crt_in`

```
std::vector<double> _crt_in
```

**12.260.4.10** `_crt_out`

```
std::vector<double> _crt_out
```

**12.260.4.11** `_size`

```
size_t _size
```

**12.260.4.12** `_pbits`

```
size_t _pbits
```

**12.260.4.13** `_ldm`

```
size_t _ldm
```

The documentation for this struct was generated from the following files:

- [rns-double.h](#)
- [rns-double.inl](#)

**12.261 RNSElementTag Struct Reference**

Representation in a Residue Number System.

```
#include <field-traits.h>
```

**12.261.1 Detailed Description**

Representation in a Residue Number System.

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

**12.262 RNSInteger< RNS > Class Template Reference**

```
#include <rns-integer.h>
```

## Data Structures

- class [RandIter](#)

## Public Types

- [typedef](#) [RNS::Element](#) [Element](#)
- [typedef](#) [RNS::Element\\_ptr](#) [Element\\_ptr](#)
- [typedef](#) [RNS::ConstElement\\_ptr](#) [ConstElement\\_ptr](#)

## Public Member Functions

- [RNSInteger](#) ([const](#) [RNS](#) &[myrns](#))
- [template](#)<[typename](#) [T](#) >  
  [RNSInteger](#) ([const](#) [T](#) &[F](#))
- [const](#) [RNS](#) & [rns](#) () [const](#)
- [size\\_t](#) [size](#) () [const](#)
- [bool](#) [isOne](#) ([const](#) [Element](#) &[x](#)) [const](#)
- [bool](#) [isMOne](#) ([const](#) [Element](#) &[x](#)) [const](#)
- [bool](#) [isZero](#) ([const](#) [Element](#) &[x](#)) [const](#)
- [integer](#) [characteristic](#) ([integer](#) &[p](#)) [const](#)
- [integer](#) [cardinality](#) ([integer](#) &[p](#)) [const](#)
- [Element](#) & [init](#) ([Element](#) &[x](#)) [const](#)
- [Element](#) & [init](#) ([Element](#) &[x](#), [const](#) [Givaro::Integer](#) &[y](#)) [const](#)
- [Element](#) & [reduce](#) ([Element](#) &[x](#), [const](#) [Element](#) &[y](#)) [const](#)
- [Element](#) & [reduce](#) ([Element](#) &[x](#)) [const](#)
- [Givaro::Integer](#) [convert](#) ([Givaro::Integer](#) &[x](#), [const](#) [Element](#) &[y](#)) [const](#)
- [Element](#) & [assign](#) ([Element](#) &[x](#), [const](#) [Element](#) &[y](#)) [const](#)
- [std::ostream](#) & [write](#) ([std::ostream](#) &[os](#), [const](#) [Element](#) &[y](#)) [const](#)
- [std::ostream](#) & [write](#) ([std::ostream](#) &[os](#)) [const](#)

## Data Fields

- [Element](#) [one](#)
- [Element](#) [mOne](#)
- [Element](#) [zero](#)

## Protected Types

- [typedef](#) [RNS::BasisElement](#) [BasisElement](#)
- [typedef](#) [Givaro::Integer](#) [integer](#)

## Protected Attributes

- [const](#) [RNS](#) \* [\\_rns](#)

## 12.262.1 Member Typedef Documentation

### 12.262.1.1 BasisElement

```
template<typename RNS >
typedef RNS::BasisElement BasisElement [protected]
```

### 12.262.1.2 integer

```
template<typename RNS >
typedef Givaro::Integer integer [protected]
```

### 12.262.1.3 Element

```
template<typename RNS >
typedef RNS::Element Element
```

### 12.262.1.4 Element\_ptr

```
template<typename RNS >
typedef RNS::Element_ptr Element_ptr
```

### 12.262.1.5 ConstElement\_ptr

```
template<typename RNS >
typedef RNS::ConstElement_ptr ConstElement_ptr
```

## 12.262.2 Constructor & Destructor Documentation

### 12.262.2.1 RNSInteger() [1/2]

```
template<typename RNS >
RNSInteger (
 const RNS & myrns) [inline]
```

### 12.262.2.2 RNSInteger() [2/2]

```
template<typename RNS >
template<typename T >
RNSInteger (
 const T & F) [inline]
```

## 12.262.3 Member Function Documentation

### 12.262.3.1 rns()

```
template<typename RNS >
const RNS & rns () const [inline]
```

### 12.262.3.2 size()

```
template<typename RNS >
size_t size () const [inline]
```

### 12.262.3.3 isOne()

```
template<typename RNS >
bool isOne (
 const Element & x) const [inline]
```

### 12.262.3.4 isMOne()

```
template<typename RNS >
bool isMOne (
 const Element & x) const [inline]
```

### 12.262.3.5 isZero()

```
template<typename RNS >
bool isZero (
 const Element & x) const [inline]
```

**12.262.3.6 characteristic()**

```
template<typename RNS >
integer characteristic (
 integer & p) const [inline]
```

**12.262.3.7 cardinality()**

```
template<typename RNS >
integer cardinality (
 integer & p) const [inline]
```

**12.262.3.8 init() [1/2]**

```
template<typename RNS >
Element & init (
 Element & x) const [inline]
```

**12.262.3.9 init() [2/2]**

```
template<typename RNS >
Element & init (
 Element & x,
 const Givaro::Integer & y) const [inline]
```

**12.262.3.10 reduce() [1/2]**

```
template<typename RNS >
Element & reduce (
 Element & x,
 const Element & y) const [inline]
```

**12.262.3.11 reduce() [2/2]**

```
template<typename RNS >
Element & reduce (
 Element & x) const [inline]
```

**12.262.3.12 convert()**

```
template<typename RNS >
Givaro::Integer convert (
 Givaro::Integer & x,
 const Element & y) const [inline]
```

**12.262.3.13 assign()**

```
template<typename RNS >
Element & assign (
 Element & x,
 const Element & y) const [inline]
```

**12.262.3.14 write() [1/2]**

```
template<typename RNS >
std::ostream & write (
 std::ostream & os,
 const Element & y) const [inline]
```

**12.262.3.15 write()** [2/2]

```
template<typename RNS >
std::ostream & write (
 std::ostream & os) const [inline]
```

**12.262.4 Field Documentation****12.262.4.1 \_rns**

```
template<typename RNS >
const RNS* _rns [protected]
```

**12.262.4.2 one**

```
template<typename RNS >
Element one
```

**12.262.4.3 mOne**

```
template<typename RNS >
Element mOne
```

**12.262.4.4 zero**

```
template<typename RNS >
Element zero
```

The documentation for this class was generated from the following files:

- [field-traits.h](#)
- [rns-integer.h](#)

**12.263 RNSIntegerMod< RNS > Class Template Reference**

```
#include <rns-integer-mod.h>
```

**Data Structures**

- class [RandIter](#)

**Public Types**

- `typedef RNS::Element Element`
- `typedef RNS::Element_ptr Element_ptr`
- `typedef RNS::ConstElement_ptr ConstElement_ptr`

**Public Member Functions**

- `RNSIntegerMod (const integer &p, const RNS &myrns)`
- `const rns_double &rns () const`
- `const RNSInteger< RNS > &delayed () const`
- `size_t size () const`
- `bool isOne (const Element &x) const`
- `bool isMOne (const Element &x) const`
- `bool isZero (const Element &x) const`
- `integer &characteristic (integer &p) const`
- `integer characteristic () const`
- `integer &cardinality (integer &p) const`
- `integer cardinality () const`

- [integer minElement \(\) const](#)
- [integer maxElement \(\) const](#)
- [Element & init \(Element &x\) const](#)
- [Element & init \(Element &x, const Givaro::Integer &y\) const](#)
- [Element & reduce \(Element &x, const Element &y\) const](#)
- [Element & reduce \(Element &x\) const](#)
- [Element & init \(Element &x, const Element &y\) const](#)
- [Givaro::Integer convert \(Givaro::Integer &x, const Element &y\) const](#)
- [Element & assign \(Element &x, const Element &y\) const](#)
- [Element & add \(Element &x, const Element &y, const Element &z\) const](#)
- [Element & sub \(Element &x, const Element &y, const Element &z\) const](#)
- [Element & neg \(Element &x, const Element &y\) const](#)
- [Element & mul \(Element &x, const Element &y, const Element &z\) const](#)
- [Element & axpyin \(Element &x, const Element &y, const Element &z\) const](#)
- [Element & inv \(Element &x, const Element &y\) const](#)
- [bool areEqual \(const Element &x, const Element &y\) const](#)
- [std::ostream & write \(std::ostream &os, const Element &y\) const](#)
- [std::ostream & write \(std::ostream &os\) const](#)
- [void reduce\\_modp \(size\\_t n, Element\\_ptr B\) const](#)
- [std::ostream & write\\_matrix \(std::ostream &c, const double \\*E, int n, int m, int lda\) const](#)
- [std::ostream & write\\_matrix\\_long \(std::ostream &c, const double \\*E, int n, int m, int lda\) const](#)
- [void reduce\\_modp \(size\\_t m, size\\_t n, Element\\_ptr B, size\\_t lda\) const](#)
- [void reduce\\_modp\\_rnsmajor \(size\\_t n, Element\\_ptr B\) const](#)

## Data Fields

- [Element one](#)
- [Element mOne](#)
- [Element zero](#)

## Protected Types

- [typedef RNS::BasisElement BasisElement](#)
- [typedef Givaro::Modular< BasisElement > ModField](#)
- [typedef Givaro::Integer integer](#)

## Protected Attributes

- [integer \\_p](#)
- [std::vector< BasisElement, AlignedAllocator< BasisElement, Alignment::CACHE\\_LINE > > \\_Mi\\_modp\\_rns](#)
- [std::vector< BasisElement, AlignedAllocator< BasisElement, Alignment::CACHE\\_LINE > > \\_iM\\_modp\\_rns](#)
- [const RNS \\* \\_rns](#)
- [Givaro::Modular< Givaro::Integer > \\_F](#)
- [RNSInteger< RNS > \\_RNSdelayed](#)

## 12.263.1 Member Typedef Documentation

### 12.263.1.1 Element

```
template<typename RNS >
typedef RNS::Element Element
```

### 12.263.1.2 Element\_ptr

```
template<typename RNS >
typedef RNS::Element_ptr Element_ptr
```

**12.263.1.3 ConstElement\_ptr**

```
template<typename RNS >
typedef RNS::ConstElement_ptr ConstElement_ptr
```

**12.263.1.4 BasisElement**

```
template<typename RNS >
typedef RNS::BasisElement BasisElement [protected]
```

**12.263.1.5 ModField**

```
template<typename RNS >
typedef Givaro::Modular<BasisElement> ModField [protected]
```

**12.263.1.6 integer**

```
template<typename RNS >
typedef Givaro::Integer integer [protected]
```

**12.263.2 Constructor & Destructor Documentation****12.263.2.1 RNSIntegerMod()**

```
template<typename RNS >
RNSIntegerMod (
 const integer & p,
 const RNS & myrns) [inline]
```

**12.263.3 Member Function Documentation****12.263.3.1 rns()**

```
template<typename RNS >
const rns_double & rns () const [inline]
```

**12.263.3.2 delayed()**

```
template<typename RNS >
const RNSInteger< RNS > & delayed () const [inline]
```

**12.263.3.3 size()**

```
template<typename RNS >
size_t size () const [inline]
```

**12.263.3.4 isOne()**

```
template<typename RNS >
bool isOne (
 const Element & x) const [inline]
```

**12.263.3.5 isMOne()**

```
template<typename RNS >
bool isMOne (
 const Element & x) const [inline]
```

**12.263.3.6 isZero()**

```
template<typename RNS >
bool isZero (
 const Element & x) const [inline]
```

**12.263.3.7 characteristic() [1/2]**

```
template<typename RNS >
integer & characteristic (
 integer & p) const [inline]
```

**12.263.3.8 characteristic() [2/2]**

```
template<typename RNS >
integer characteristic () const [inline]
```

**12.263.3.9 cardinality() [1/2]**

```
template<typename RNS >
integer & cardinality (
 integer & p) const [inline]
```

**12.263.3.10 cardinality() [2/2]**

```
template<typename RNS >
integer cardinality () const [inline]
```

**12.263.3.11 minElement()**

```
template<typename RNS >
integer minElement () const [inline]
```

**12.263.3.12 maxElement()**

```
template<typename RNS >
integer maxElement () const [inline]
```

**12.263.3.13 init() [1/3]**

```
template<typename RNS >
Element & init (
 Element & x) const [inline]
```

**12.263.3.14 init() [2/3]**

```
template<typename RNS >
Element & init (
 Element & x,
 const Givaro::Integer & y) const [inline]
```

**12.263.3.15 reduce() [1/2]**

```
template<typename RNS >
Element & reduce (
 Element & x,
 const Element & y) const [inline]
```



**12.263.3.16 reduce()** [2/2]

```
template<typename RNS >
Element & reduce (
 Element & x) const [inline]
```

**12.263.3.17 init()** [3/3]

```
template<typename RNS >
Element & init (
 Element & x,
 const Element & y) const [inline]
```

**12.263.3.18 convert()**

```
template<typename RNS >
Givaro::Integer convert (
 Givaro::Integer & x,
 const Element & y) const [inline]
```

**12.263.3.19 assign()**

```
template<typename RNS >
Element & assign (
 Element & x,
 const Element & y) const [inline]
```

**12.263.3.20 add()**

```
template<typename RNS >
Element & add (
 Element & x,
 const Element & y,
 const Element & z) const [inline]
```

**12.263.3.21 sub()**

```
template<typename RNS >
Element & sub (
 Element & x,
 const Element & y,
 const Element & z) const [inline]
```

**12.263.3.22 neg()**

```
template<typename RNS >
Element & neg (
 Element & x,
 const Element & y) const [inline]
```

**12.263.3.23 mul()**

```
template<typename RNS >
Element & mul (
 Element & x,
 const Element & y,
 const Element & z) const [inline]
```

**12.263.3.24 axpyin()**

```
template<typename RNS >
Element & axpyin (
 Element & x,
 const Element & y,
 const Element & z) const [inline]
```

**12.263.3.25 inv()**

```
template<typename RNS >
Element & inv (
 Element & x,
 const Element & y) const [inline]
```

**12.263.3.26 areEqual()**

```
template<typename RNS >
bool areEqual (
 const Element & x,
 const Element & y) const [inline]
```

**12.263.3.27 write() [1/2]**

```
template<typename RNS >
std::ostream & write (
 std::ostream & os,
 const Element & y) const [inline]
```

**12.263.3.28 write() [2/2]**

```
template<typename RNS >
std::ostream & write (
 std::ostream & os) const [inline]
```

**12.263.3.29 reduce\_modp() [1/2]**

```
template<typename RNS >
void reduce_modp (
 size_t n,
 Element_ptr B) const [inline]
```

**12.263.3.30 write\_matrix()**

```
template<typename RNS >
std::ostream & write_matrix (
 std::ostream & c,
 const double * E,
 int n,
 int m,
 int lda) const [inline]
```

**12.263.3.31 write\_matrix\_long()**

```
template<typename RNS >
std::ostream & write_matrix_long (
 std::ostream & c,
 const double * E,
 int n,
```

```

 int m,
 int lda) const [inline]

```

### 12.263.3.32 reduce\_modp() [2/2]

```

template<typename RNS >
void reduce_modp (
 size_t m,
 size_t n,
 Element_ptr B,
 size_t lda) const [inline]

```

### 12.263.3.33 reduce\_modp\_rnsmajor()

```

template<typename RNS >
void reduce_modp_rnsmajor (
 size_t n,
 Element_ptr B) const [inline]

```

## 12.263.4 Field Documentation

### 12.263.4.1 \_p

```

template<typename RNS >
integer _p [protected]

```

### 12.263.4.2 \_Mi\_modp\_rns

```

template<typename RNS >
std::vector<BasisElement, AlignedAllocator<BasisElement, Alignment::CACHE_LINE> > _Mi_modp_↵
rns [protected]

```

### 12.263.4.3 \_iM\_modp\_rns

```

template<typename RNS >
std::vector<BasisElement, AlignedAllocator<BasisElement, Alignment::CACHE_LINE> > _iM_modp_↵
rns [protected]

```

### 12.263.4.4 \_rns

```

template<typename RNS >
const RNS* _rns [protected]

```

### 12.263.4.5 \_F

```

template<typename RNS >
Givaro::Modular<Givaro::Integer> _F [protected]

```

### 12.263.4.6 \_RNSdelayed

```

template<typename RNS >
RNSInteger<RNS> _RNSdelayed [protected]

```

### 12.263.4.7 one

```

template<typename RNS >
Element one

```

#### 12.263.4.8 mOne

```
template<typename RNS >
Element mOne
```

#### 12.263.4.9 zero

```
template<typename RNS >
Element zero
```

The documentation for this class was generated from the following files:

- [field-traits.h](#)
- [rns-integer-mod.h](#)

## 12.264 rnsRandIter< RNS > Class Template Reference

```
#include <rns-double.h>
```

Inheritance diagram for rnsRandIter< RNS >:



### Public Member Functions

- [rnsRandIter](#) (const [RNS](#) &R, [uint64\\_t](#) seed=0)
- [RNS::Element](#) & random (typename [RNS::Element](#) &elt) const  
*RNS ring Element random assignement.*
- [RNS::Element](#) & operator() (typename [RNS::Element](#) &elt) const
- [RNS::Element](#) operator() () const
- [RNS::Element](#) random () const
- const [RNS](#) & ring () const

### 12.264.1 Constructor & Destructor Documentation

#### 12.264.1.1 rnsRandIter()

```
template<typename RNS >
rnsRandIter (
 const RNS & R,
 uint64_t seed = 0) [inline]
```

### 12.264.2 Member Function Documentation

#### 12.264.2.1 random() [1/2]

```
template<typename RNS >
RNS::Element & random (
 typename RNS::Element & elt) const [inline]
```

RNS ring Element random assignement.

Element is supposed to be initialized

#### Returns

random ring Element

**12.264.2.2 operator>() [1/2]**

```
template<typename RNS >
RNS::Element & operator() (
 typename RNS::Element & elt) const [inline]
```

**12.264.2.3 operator>() [2/2]**

```
template<typename RNS >
RNS::Element operator() () const [inline]
```

**12.264.2.4 random() [2/2]**

```
template<typename RNS >
RNS::Element random () const [inline]
```

**12.264.2.5 ring()**

```
template<typename RNS >
const RNS & ring () const [inline]
```

The documentation for this class was generated from the following file:

- [rns-double.h](#)

**12.265 Row Struct Reference**

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

**12.266 rint< K > Class Template Reference**

The documentation for this class was generated from the following file:

- [field-traits.h](#)

**12.267 ScalFunctions< Element > Struct Template Reference**

Inheritance diagram for ScalFunctions< Element >:

**Public Types**

- [using vectElt](#) = vector< Element >

**Static Public Member Functions**

- [static void genInputs](#) (vector< [vectElt](#) > &inputs)
- [static void genInputsWithZero](#) (vector< [vectElt](#) > &inputs)
- [static Element zero](#) ()
- [static Element vand](#) (Element [x1](#), Element [x2](#))
- [static Element vor](#) (Element [x1](#), Element [x2](#))

- `static` Element `vxor` (Element `x1`, Element `x2`)
- `static` Element `vandnot` (Element `x1`, Element `x2`)
- `static` Element `add` (Element `x1`, Element `x2`)
- `static` Element `addin` (Element `&x1`, Element `x2`)
- `static` Element `sub` (Element `x1`, Element `x2`)
- `static` Element `subin` (Element `&x1`, Element `x2`)
- `static` Element `mul` (Element `x1`, Element `x2`)
- `static` Element `mulin` (Element `&x1`, Element `x2`)
- `static` Element `div` (Element `x1`, Element `x2`)
- `static` Element `fmadd` (Element `x1`, Element `x2`, Element `x3`)
- `static` Element `fmaddin` (Element `&x1`, Element `x2`, Element `x3`)
- `static` Element `fmsub` (Element `x1`, Element `x2`, Element `x3`)
- `static` Element `fmsubin` (Element `&x1`, Element `x2`, Element `x3`)
- `static` Element `fnmadd` (Element `x1`, Element `x2`, Element `x3`)
- `static` Element `fnmaddin` (Element `&x1`, Element `x2`, Element `x3`)
- `static` Element `lesser` (Element `x1`, Element `x2`)
- `static` Element `lesser_eq` (Element `x1`, Element `x2`)
- `static` Element `greater` (Element `x1`, Element `x2`)
- `static` Element `greater_eq` (Element `x1`, Element `x2`)
- `static` Element `eq` (Element `x1`, Element `x2`)
- `static` vectElt `unpacklo` (vectElt `a`, vectElt `b`)
- `static` vectElt `unpackhi` (vectElt `a`, vectElt `b`)
- `static` void `unpacklohi` (vectElt `&lo`, vectElt `&hi`, vectElt `a`, vectElt `b`)
- `static` vectElt `pack_even` (vectElt `a`, vectElt `b`)
- `static` vectElt `pack_odd` (vectElt `a`, vectElt `b`)
- `static` void `pack` (vectElt `&even`, vectElt `&odd`, vectElt `a`, vectElt `b`)
- `template<uint16_t s>`  
`static` vectElt `blend` (vectElt `a`, vectElt `b`)

## 12.267.1 Member Typedef Documentation

### 12.267.1.1 vectElt

```
template<typename Element >
using vectElt = vector<Element>
```

## 12.267.2 Member Function Documentation

### 12.267.2.1 genInputs()

```
template<typename Element >
static void genInputs (
 vector< vectElt > & inputs) [inline], [static]
```

### 12.267.2.2 genInputsWithZero()

```
template<typename Element >
static void genInputsWithZero (
 vector< vectElt > & inputs) [inline], [static]
```

### 12.267.2.3 zero()

```
template<typename Element >
static Element zero () [inline], [static]
```

#### 12.267.2.4 vand()

```
template<typename Element >
static Element vand (
 Element x1,
 Element x2) [inline], [static]
```

#### 12.267.2.5 vor()

```
template<typename Element >
static Element vor (
 Element x1,
 Element x2) [inline], [static]
```

#### 12.267.2.6 vxor()

```
template<typename Element >
static Element vxor (
 Element x1,
 Element x2) [inline], [static]
```

#### 12.267.2.7 vandnot()

```
template<typename Element >
static Element vandnot (
 Element x1,
 Element x2) [inline], [static]
```

#### 12.267.2.8 add()

```
template<typename Element >
static Element add (
 Element x1,
 Element x2) [inline], [static]
```

#### 12.267.2.9 addin()

```
template<typename Element >
static Element addin (
 Element & x1,
 Element x2) [inline], [static]
```

#### 12.267.2.10 sub()

```
template<typename Element >
static Element sub (
 Element x1,
 Element x2) [inline], [static]
```

#### 12.267.2.11 subin()

```
template<typename Element >
static Element subin (
 Element & x1,
 Element x2) [inline], [static]
```

#### 12.267.2.12 mul()

```
template<typename Element >
static Element mul (
```

```
 Element x1,
 Element x2) [inline], [static]
```

#### 12.267.2.13 mulin()

```
template<typename Element >
static Element mulin (
 Element & x1,
 Element x2) [inline], [static]
```

#### 12.267.2.14 div()

```
template<typename Element >
static Element div (
 Element x1,
 Element x2) [inline], [static]
```

#### 12.267.2.15 fmadd()

```
template<typename Element >
static Element fmadd (
 Element x1,
 Element x2,
 Element x3) [inline], [static]
```

#### 12.267.2.16 fmaddin()

```
template<typename Element >
static Element fmaddin (
 Element & x1,
 Element x2,
 Element x3) [inline], [static]
```

#### 12.267.2.17 fmsub()

```
template<typename Element >
static Element fmsub (
 Element x1,
 Element x2,
 Element x3) [inline], [static]
```

#### 12.267.2.18 fmsubin()

```
template<typename Element >
static Element fmsubin (
 Element & x1,
 Element x2,
 Element x3) [inline], [static]
```

#### 12.267.2.19 fnmadd()

```
template<typename Element >
static Element fnmadd (
 Element x1,
 Element x2,
 Element x3) [inline], [static]
```



**12.267.2.20 fnmaddin()**

```
template<typename Element >
static Element fnmaddin (
 Element & x1,
 Element x2,
 Element x3) [inline], [static]
```

**12.267.2.21 lesser()**

```
template<typename Element >
static Element lesser (
 Element x1,
 Element x2) [inline], [static]
```

**12.267.2.22 lesser\_eq()**

```
template<typename Element >
static Element lesser_eq (
 Element x1,
 Element x2) [inline], [static]
```

**12.267.2.23 greater()**

```
template<typename Element >
static Element greater (
 Element x1,
 Element x2) [inline], [static]
```

**12.267.2.24 greater\_eq()**

```
template<typename Element >
static Element greater_eq (
 Element x1,
 Element x2) [inline], [static]
```

**12.267.2.25 eq()**

```
template<typename Element >
static Element eq (
 Element x1,
 Element x2) [inline], [static]
```

**12.267.2.26 unpacklo()**

```
template<typename Element >
static vectElt unpacklo (
 vectElt a,
 vectElt b) [inline], [static]
```

**12.267.2.27 unpackhi()**

```
template<typename Element >
static vectElt unpackhi (
 vectElt a,
 vectElt b) [inline], [static]
```

**12.267.2.28 unpacklohi()**

```
template<typename Element >
static void unpacklohi (
 vectElt & lo,
 vectElt & hi,
 vectElt a,
 vectElt b) [inline], [static]
```

**12.267.2.29 pack\_even()**

```
template<typename Element >
static vectElt pack_even (
 vectElt a,
 vectElt b) [inline], [static]
```

**12.267.2.30 pack\_odd()**

```
template<typename Element >
static vectElt pack_odd (
 vectElt a,
 vectElt b) [inline], [static]
```

**12.267.2.31 pack()**

```
template<typename Element >
static void pack (
 vectElt & even,
 vectElt & odd,
 vectElt a,
 vectElt b) [inline], [static]
```

**12.267.2.32 blend()**

```
template<typename Element >
template<uint16_t s>
static vectElt blend (
 vectElt a,
 vectElt b) [inline], [static]
```

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.268 ScalFunctionsBase< Element, Enable > Struct Template Reference

Inheritance diagram for ScalFunctionsBase< Element, Enable >:



The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.269 ScalFunctionsBase< Element, typename enable\_if< is\_floating\_point< Element >::value >::type > Struct Template Reference

### Data Structures

- class [FloatingPointTestDistribution](#)

### Static Public Member Functions

- [static](#) [FloatingPointTestDistribution](#) [get\\_default\\_random\\_generator](#) ()
- [static](#) [Element](#) [ceil](#) ([Element](#) [x](#))
- [static](#) [Element](#) [floor](#) ([Element](#) [x](#))
- [static](#) [Element](#) [round](#) ([Element](#) [x](#))
- [static](#) [Element](#) [blendv](#) ([Element](#) [a](#), [Element](#) [b](#), [Element](#) [mask](#))
- [static](#) [Element](#) [fma](#) ([Element](#) [x](#), [Element](#) [y](#), [Element](#) [z](#))

### Static Public Attributes

- [static constexpr](#) [Element](#) [\\_zero](#) = 0.0
- [static constexpr](#) [Element](#) [cmp\\_true](#) = NAN
- [static constexpr](#) [Element](#) [cmp\\_false](#) = [\\_zero](#)

## 12.269.1 Member Function Documentation

### 12.269.1.1 [get\\_default\\_random\\_generator\(\)](#)

```
template<typename Element >
static FloatingPointTestDistribution get_default_random_generator () [inline], [static]
```

### 12.269.1.2 [ceil\(\)](#)

```
template<typename Element >
static Element ceil (
 Element x) [inline], [static]
```

### 12.269.1.3 [floor\(\)](#)

```
template<typename Element >
static Element floor (
 Element x) [inline], [static]
```

### 12.269.1.4 [round\(\)](#)

```
template<typename Element >
static Element round (
 Element x) [inline], [static]
```

### 12.269.1.5 [blendv\(\)](#)

```
template<typename Element >
static Element blendv (
 Element a,
 Element b,
 Element mask) [inline], [static]
```

### 12.269.1.6 fma()

```
template<typename Element >
static Element fma (
 Element x,
 Element y,
 Element z) [inline], [static]
```

## 12.269.2 Field Documentation

### 12.269.2.1 \_zero

```
template<typename Element >
constexpr Element _zero = 0.0 [static], [constexpr]
```

### 12.269.2.2 cmp\_true

```
template<typename Element >
constexpr Element cmp_true = NAN [static], [constexpr]
```

### 12.269.2.3 cmp\_false

```
template<typename Element >
constexpr Element cmp_false = _zero [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.270 ScalFunctionsBase< Element, typename enable\_if< is\_integral< Element >::value >::type > Struct Template Reference

### Static Public Member Functions

- [static](#) [uniform\\_int\\_distribution](#)< Element > [get\\_default\\_random\\_generator](#) ()
- [static](#) Element [round](#) (Element x)
- [static](#) Element [fma](#) (Element x, Element y, Element z)
- [static](#) Element [mullo](#) (Element x1, Element x2)
- [static](#) Element [mulhi](#) (Element x1, Element x2)
- [static](#) Element [mulx](#) (Element x1, Element x2)
- [static](#) Element [fmaddx](#) (Element x1, Element x2, Element x3)
- [static](#) Element [fmaddxin](#) (Element &x1, Element x2, Element x3)
- [static](#) Element [fmsubx](#) (Element x1, Element x2, Element x3)
- [static](#) Element [fmsubxin](#) (Element &x1, Element x2, Element x3)
- [static](#) Element [fnmaddx](#) (Element x1, Element x2, Element x3)
- [static](#) Element [fnmaddxin](#) (Element &x1, Element x2, Element x3)
- [template](#)<int s>  
  [static](#) Element [sra](#) (Element x1)
- [template](#)<int s>  
  [static](#) Element [srl](#) (Element x1)
- [template](#)<int s>  
  [static](#) Element [sll](#) (Element x1)

### Static Public Attributes

- [static constexpr](#) Element [\\_zero](#) = 0
- [static constexpr](#) Element [cmp\\_true](#) = -1
- [static constexpr](#) Element [cmp\\_false](#) = [\\_zero](#)

## 12.270.1 Member Function Documentation

### 12.270.1.1 get\_default\_random\_generator()

```
template<typename Element >
static uniform_int_distribution< Element > get_default_random_generator () [inline], [static]
```

### 12.270.1.2 round()

```
template<typename Element >
static Element round (
 Element x) [inline], [static]
```

### 12.270.1.3 fma()

```
template<typename Element >
static Element fma (
 Element x,
 Element y,
 Element z) [inline], [static]
```

### 12.270.1.4 mullo()

```
template<typename Element >
static Element mullo (
 Element x1,
 Element x2) [inline], [static]
```

### 12.270.1.5 mulhi()

```
template<typename Element >
static Element mulhi (
 Element x1,
 Element x2) [inline], [static]
```

### 12.270.1.6 mulx()

```
template<typename Element >
static Element mulx (
 Element x1,
 Element x2) [inline], [static]
```

### 12.270.1.7 fmaddx()

```
template<typename Element >
static Element fmaddx (
 Element x1,
 Element x2,
 Element x3) [inline], [static]
```

### 12.270.1.8 fmaddxin()

```
template<typename Element >
static Element fmaddxin (
 Element & x1,
 Element x2,
 Element x3) [inline], [static]
```

**12.270.1.9 fmsubx()**

```
template<typename Element >
static Element fmsubx (
 Element x1,
 Element x2,
 Element x3) [inline], [static]
```

**12.270.1.10 fmsubxin()**

```
template<typename Element >
static Element fmsubxin (
 Element & x1,
 Element x2,
 Element x3) [inline], [static]
```

**12.270.1.11 fnmaddx()**

```
template<typename Element >
static Element fnmaddx (
 Element x1,
 Element x2,
 Element x3) [inline], [static]
```

**12.270.1.12 fnmaddxin()**

```
template<typename Element >
static Element fnmaddxin (
 Element & x1,
 Element x2,
 Element x3) [inline], [static]
```

**12.270.1.13 sra()**

```
template<typename Element >
template<int s>
static Element sra (
 Element x1) [inline], [static]
```

**12.270.1.14 srl()**

```
template<typename Element >
template<int s>
static Element srl (
 Element x1) [inline], [static]
```

**12.270.1.15 sll()**

```
template<typename Element >
template<int s>
static Element sll (
 Element x1) [inline], [static]
```

**12.270.2 Field Documentation****12.270.2.1 \_zero**

```
template<typename Element >
constexpr Element _zero = 0 [static], [constexpr]
```

**12.270.2.2 cmp\_true**

```
template<typename Element >
constexpr Element cmp_true = -1 [static], [constexpr]
```

**12.270.2.3 cmp\_false**

```
template<typename Element >
constexpr Element cmp_false = _zero [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

**12.271 Sequential Struct Reference****Public Member Functions**

- [Sequential](#) ()
- [Sequential](#) (size\_t nth)
- template<class Cut, class Param >  
  [Sequential](#) (Parallel< Cut, Param > &)
- size\_t numthreads () const

**Friends**

- std::ostream & [operator<<](#) (std::ostream &out, const [Sequential](#) &)

**12.271.1 Constructor & Destructor Documentation****12.271.1.1 Sequential() [1/3]**

```
Sequential () [inline]
```

**12.271.1.2 Sequential() [2/3]**

```
Sequential (
 size_t nth) [inline]
```

**12.271.1.3 Sequential() [3/3]**

```
template<class Cut, class Param >
Sequential (
 Parallel< Cut, Param > &) [inline]
```

**12.271.2 Member Function Documentation****12.271.2.1 numthreads()**

```
size_t numthreads () const [inline]
```

**12.271.3 Friends And Related Symbol Documentation****12.271.3.1 operator<<**

```
std::ostream & operator<< (
 std::ostream & out,
 const Sequential &) [friend]
```

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

## 12.272 Simd128\_impl< ArithType, Int, Signed, Size > Struct Template Reference

The documentation for this struct was generated from the following file:

- [simd128.inl](#)

## 12.273 Simd128\_impl< true, false, true, 4 > Struct Reference

The documentation for this struct was generated from the following file:

- [simd128\\_float.inl](#)

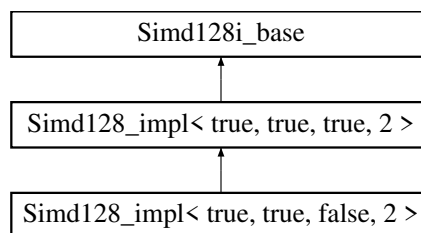
## 12.274 Simd128\_impl< true, false, true, 8 > Struct Reference

The documentation for this struct was generated from the following file:

- [simd128\\_double.inl](#)

## 12.275 Simd128\_impl< true, true, false, 2 > Struct Reference

Inheritance diagram for Simd128\_impl< true, true, false, 2 >:



### Data Structures

- union [Converter](#)

### Public Types

- using [scalar\\_t](#) = [uint16\\_t](#)
- using [aligned\\_allocator](#) = [AlignedAllocator](#)< [scalar\\_t](#), [Alignment](#)([alignment](#))>
- using [aligned\\_vector](#) = [std::vector](#)< [scalar\\_t](#), [aligned\\_allocator](#) >
- template<class [Field](#) >  
using [is\\_same\\_element](#) = [std::is\\_same](#)< [typename Field::Element](#), [scalar\\_t](#) >
- using [vect\\_t](#) = [\\_\\_m128i](#)

### Static Public Member Functions

- static const [std::string](#) [type\\_string](#) ()
- static [INLINE CONST](#) [vect\\_t](#) [set1](#) ([const scalar\\_t](#) x)
- static [INLINE CONST](#) [vect\\_t](#) [set](#) ([const scalar\\_t](#) x0, [const scalar\\_t](#) x1, [const scalar\\_t](#) x2, [const scalar\\_t](#) x3, [const scalar\\_t](#) x4, [const scalar\\_t](#) x5, [const scalar\\_t](#) x6, [const scalar\\_t](#) x7)
- template<class [T](#) >  
static [INLINE PURE](#) [vect\\_t](#) [gather](#) ([const scalar\\_t](#) \*[const](#) p, [const T](#) \*[const](#) idx)
- static [INLINE PURE](#) [vect\\_t](#) [load](#) ([const scalar\\_t](#) \*[const](#) p)
- static [INLINE PURE](#) [vect\\_t](#) [loadu](#) ([const scalar\\_t](#) \*[const](#) p)
- static [INLINE](#) void [store](#) ([scalar\\_t](#) \*p, [vect\\_t](#) v)



- static INLINE void storeu (scalar\_t \*p, vect\_t v)
- static INLINE void stream (scalar\_t \*p, const vect\_t v)
- template<int s>
  - static INLINE CONST vect\_t sra (const vect\_t a)
- static INLINE CONST vect\_t greater (vect\_t a, vect\_t b)
- static INLINE CONST vect\_t lesser (vect\_t a, vect\_t b)
- static INLINE CONST vect\_t greater\_eq (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t lesser\_eq (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t mulhi (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t mulx (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fmadddx (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fmadddx (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fnmadddx (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fnmadddx (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fmsubx (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fmsubx (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST scalar\_t hadd\_to\_scal (const vect\_t a)
- template<class T>
  - static constexpr bool valid (T \*p)
- template<class T>
  - static constexpr bool compliant (T n)
- static INLINE CONST vect\_t set1 (const scalar\_t x)
- static INLINE CONST vect\_t set (const scalar\_t x0, const scalar\_t x1, const scalar\_t x2, const scalar\_t x3, const scalar\_t x4, const scalar\_t x5, const scalar\_t x6, const scalar\_t x7)
- template<class T>
  - static INLINE PURE vect\_t gather (const scalar\_t \*const p, const T \*const idx)
- static INLINE PURE vect\_t load (const scalar\_t \*const p)
- static INLINE PURE vect\_t loadu (const scalar\_t \*const p)
- static INLINE void store (scalar\_t \*p, vect\_t v)
- static INLINE void storeu (scalar\_t \*p, vect\_t v)
- static INLINE void stream (scalar\_t \*p, const vect\_t v)
- template<int s>
  - static INLINE CONST vect\_t sll (const vect\_t a)
- template<int s>
  - static INLINE CONST vect\_t srl (const vect\_t a)
- template<uint32\_t s>
  - static INLINE CONST vect\_t shuffle (const vect\_t a)
- static INLINE CONST vect\_t unpacklo\_intrinsic (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t unpackhi\_intrinsic (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t unpacklo (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t unpackhi (const vect\_t a, const vect\_t b)
- static INLINE void unpacklohi (vect\_t &lo, vect\_t &hi, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t pack\_even (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t pack\_odd (const vect\_t a, const vect\_t b)
- static INLINE void pack (vect\_t &even, vect\_t &odd, const vect\_t a, const vect\_t b)
- static INLINE void transpose (vect\_t &r0, vect\_t &r1, vect\_t &r2, vect\_t &r3, vect\_t &r4, vect\_t &r5, vect\_t &r6, vect\_t &r7)
- template<uint8\_t s>
  - static INLINE CONST vect\_t blend (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t add (const vect\_t a, const vect\_t b)
- static INLINE vect\_t addin (vect\_t &a, const vect\_t b)
- static INLINE CONST vect\_t sub (const vect\_t a, const vect\_t b)
- static INLINE vect\_t subin (vect\_t &a, const vect\_t b)
- static INLINE CONST vect\_t mullo (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t mul (const vect\_t a, const vect\_t b)

- `static inline const vect_t fmadd (const vect_t c, const vect_t a, const vect_t b)`
- `static inline vect_t fmaddin (vect_t &c, const vect_t a, const vect_t b)`
- `static inline const vect_t fnmadd (const vect_t c, const vect_t a, const vect_t b)`
- `static inline vect_t fnmaddin (vect_t &c, const vect_t a, const vect_t b)`
- `static inline const vect_t fmsub (const vect_t c, const vect_t a, const vect_t b)`
- `static inline vect_t fmsubin (vect_t &c, const vect_t a, const vect_t b)`
- `static inline const vect_t eq (const vect_t a, const vect_t b)`
- `static inline const vect_t round (const vect_t a)`
- `static inline vect_t mod (vect_t &C, const vect_t &P, const __m64 &INVP, const vect_t &NEGP, const vect_t &MIN, const vect_t &MAX, vect_t &Q, vect_t &T)`
- `static inline const vect_t zero ()`
- `template<uint8_t s>`  
`static inline const vect_t sll128 (const vect_t a)`
- `template<uint8_t s>`  
`static inline const vect_t srl128 (const vect_t a)`
- `static inline const vect_t vand (const vect_t a, const vect_t b)`
- `static inline const vect_t vor (const vect_t a, const vect_t b)`
- `static inline const vect_t vxor (const vect_t a, const vect_t b)`
- `static inline const vect_t vandnot (const vect_t a, const vect_t b)`

### Static Public Attributes

- `static const constexpr size_t vect_size = 8`
- `static const constexpr size_t alignment = 16`

## 12.275.1 Member Typedef Documentation

### 12.275.1.1 scalar\_t

```
using scalar_t = uint16_t
```

### 12.275.1.2 aligned\_allocator

```
using aligned_allocator = AlignedAllocator<scalar_t, Alignment(alignment)>
```

### 12.275.1.3 aligned\_vector

```
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```

### 12.275.1.4 is\_same\_element

```
template<class Field >
using is_same_element = std::is_same<typename Field::Element, scalar_t>
```

### 12.275.1.5 vect\_t

```
using vect_t = __m128i [inherited]
```

## 12.275.2 Member Function Documentation

### 12.275.2.1 type\_string()

```
static const std::string type_string () [inline], [static]
```

### 12.275.2.2 set1() [1/2]

```
static inline const vect_t set1 (
 const scalar_t x) [inline], [static]
```

**12.275.2.3 set()** [1/2]

```
static INLINE CONST vect_t set (
 const scalar_t x0,
 const scalar_t x1,
 const scalar_t x2,
 const scalar_t x3,
 const scalar_t x4,
 const scalar_t x5,
 const scalar_t x6,
 const scalar_t x7) [inline], [static]
```

**12.275.2.4 gather()** [1/2]

```
template<class T >
static INLINE PURE vect_t gather (
 const scalar_t *const p,
 const T *const idx) [inline], [static]
```

**12.275.2.5 load()** [1/2]

```
static INLINE PURE vect_t load (
 const scalar_t *const p) [inline], [static]
```

**12.275.2.6 loadu()** [1/2]

```
static INLINE PURE vect_t loadu (
 const scalar_t *const p) [inline], [static]
```

**12.275.2.7 store()** [1/2]

```
static INLINE void store (
 scalar_t * p,
 vect_t v) [inline], [static]
```

**12.275.2.8 storeu()** [1/2]

```
static INLINE void storeu (
 scalar_t * p,
 vect_t v) [inline], [static]
```

**12.275.2.9 stream()** [1/2]

```
static INLINE void stream (
 scalar_t * p,
 const vect_t v) [inline], [static]
```

**12.275.2.10 sra()**

```
template<int s>
static INLINE CONST vect_t sra (
 const vect_t a) [inline], [static]
```

**12.275.2.11 greater()**

```
static INLINE CONST vect_t greater (
 vect_t a,
 vect_t b) [inline], [static]
```

**12.275.2.12 lesser()**

```
static INLINE CONST vect_t lesser (
 vect_t a,
 vect_t b) [inline], [static]
```

**12.275.2.13 greater\_eq()**

```
static INLINE CONST vect_t greater_eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.275.2.14 lesser\_eq()**

```
static INLINE CONST vect_t lesser_eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.275.2.15 mulhi()**

```
static INLINE CONST vect_t mulhi (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.275.2.16 mulx()**

```
static INLINE CONST vect_t mulx (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.275.2.17 fmaddx()**

```
static INLINE CONST vect_t fmaddx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.275.2.18 fmaddxin()**

```
static INLINE vect_t fmaddxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.275.2.19 fnmaddx()**

```
static INLINE CONST vect_t fnmaddx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.275.2.20 fnmaddxin()**

```
static INLINE vect_t fnmaddxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.275.2.21 fmsubx()**

```
static INLINE CONST vect_t fmsubx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.275.2.22 fmsubxin()**

```
static INLINE vect_t fmsubxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.275.2.23 hadd\_to\_scal()**

```
static INLINE CONST scalar_t hadd_to_scal (
 const vect_t a) [inline], [static]
```

**12.275.2.24 valid()**

```
template<class T >
static constexpr bool valid (
 T * p) [inline], [static], [constexpr], [inherited]
```

**12.275.2.25 compliant()**

```
template<class T >
static constexpr bool compliant (
 T n) [inline], [static], [constexpr], [inherited]
```

**12.275.2.26 set1() [2/2]**

```
static INLINE CONST vect_t set1 (
 const scalar_t x) [inline], [static], [inherited]
```

**12.275.2.27 set() [2/2]**

```
static INLINE CONST vect_t set (
 const scalar_t x0,
 const scalar_t x1,
 const scalar_t x2,
 const scalar_t x3,
 const scalar_t x4,
 const scalar_t x5,
 const scalar_t x6,
 const scalar_t x7) [inline], [static], [inherited]
```

**12.275.2.28 gather() [2/2]**

```
template<class T >
static INLINE PURE vect_t gather (
 const scalar_t *const p,
 const T *const idx) [inline], [static], [inherited]
```

**12.275.2.29 load() [2/2]**

```
static INLINE PURE vect_t load (
 const scalar_t *const p) [inline], [static], [inherited]
```

**12.275.2.30 loadu() [2/2]**

```
static INLINE PURE vect_t loadu (
 const scalar_t *const p) [inline], [static], [inherited]
```

**12.275.2.31 store() [2/2]**

```
static INLINE void store (
 scalar_t * p,
 vect_t v) [inline], [static], [inherited]
```

**12.275.2.32 storeu() [2/2]**

```
static INLINE void storeu (
 scalar_t * p,
 vect_t v) [inline], [static], [inherited]
```

**12.275.2.33 stream() [2/2]**

```
static INLINE void stream (
 scalar_t * p,
 const vect_t v) [inline], [static], [inherited]
```

**12.275.2.34 sll()**

```
template<int s>
static INLINE CONST vect_t sll (
 const vect_t a) [inline], [static], [inherited]
```

**12.275.2.35 srl()**

```
template<int s>
static INLINE CONST vect_t srl (
 const vect_t a) [inline], [static], [inherited]
```

**12.275.2.36 shuffle()**

```
template<uint32_t s>
static INLINE CONST vect_t shuffle (
 const vect_t a) [inline], [static], [inherited]
```

**12.275.2.37 unpacklo\_intrinsic()**

```
static INLINE CONST vect_t unpacklo_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.275.2.38 unpackhi\_intrinsic()**

```
static INLINE CONST vect_t unpackhi_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.275.2.39 unpacklo()**

```
static INLINE CONST vect_t unpacklo (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.275.2.40 unpackhi()**

```
static INLINE CONST vect_t unpackhi (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.275.2.41 unpacklohi()**

```
static INLINE void unpacklohi (
 vect_t & lo,
 vect_t & hi,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.275.2.42 pack\_even()**

```
static INLINE CONST vect_t pack_even (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.275.2.43 pack\_odd()**

```
static INLINE CONST vect_t pack_odd (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.275.2.44 pack()**

```
static INLINE void pack (
 vect_t & even,
 vect_t & odd,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.275.2.45 transpose()**

```
static INLINE void transpose (
 vect_t & r0,
 vect_t & r1,
 vect_t & r2,
 vect_t & r3,
 vect_t & r4,
 vect_t & r5,
 vect_t & r6,
 vect_t & r7) [inline], [static], [inherited]
```

**12.275.2.46 blend()**

```
template<uint8_t s>
static INLINE CONST vect_t blend (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.275.2.47 add()**

```
static INLINE CONST vect_t add (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.275.2.48 addin()**

```
static INLINE vect_t addin (
 vect_t & a,
 const vect_t b) [inline], [static], [inherited]
```

**12.275.2.49 sub()**

```
static INLINE CONST vect_t sub (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.275.2.50 subin()**

```
static INLINE vect_t subin (
 vect_t & a,
 const vect_t b) [inline], [static], [inherited]
```

**12.275.2.51 mullo()**

```
static INLINE CONST vect_t mullo (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.275.2.52 mul()**

```
static INLINE CONST vect_t mul (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.275.2.53 fmadd()**

```
static INLINE CONST vect_t fmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.275.2.54 fmaddin()**

```
static INLINE vect_t fmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.275.2.55 fnmadd()**

```
static INLINE CONST vect_t fnmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.275.2.56 fnmaddin()**

```
static INLINE vect_t fnmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```



**12.275.2.57 fmsub()**

```
static INLINE CONST vect_t fmsub (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.275.2.58 fmsubin()**

```
static INLINE vect_t fmsubin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.275.2.59 eq()**

```
static INLINE CONST vect_t eq (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.275.2.60 round()**

```
static INLINE CONST vect_t round (
 const vect_t a) [inline], [static], [inherited]
```

**12.275.2.61 mod()**

```
static INLINE vect_t mod (
 vect_t & C,
 const vect_t & P,
 const __m64 & INVP,
 const vect_t & NEGP,
 const vect_t & MIN,
 const vect_t & MAX,
 vect_t & Q,
 vect_t & T) [inline], [static], [inherited]
```

**12.275.2.62 zero()**

```
static INLINE CONST vect_t zero () [inline], [static], [inherited]
```

**12.275.2.63 sll128()**

```
template<uint8_t s>
static INLINE CONST vect_t sll128 (
 const vect_t a) [inline], [static], [inherited]
```

**12.275.2.64 srl128()**

```
template<uint8_t s>
static INLINE CONST vect_t srl128 (
 const vect_t a) [inline], [static], [inherited]
```

**12.275.2.65 vand()**

```
static INLINE CONST vect_t vand (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.275.2.66 vor()**

```
static INLINE CONST vect_t vor (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.275.2.67 vxor()**

```
static INLINE CONST vect_t vxor (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.275.2.68 vandnot()**

```
static INLINE CONST vect_t vandnot (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.275.3 Field Documentation****12.275.3.1 vect\_size**

```
const constexpr size_t vect_size = 8 [static], [constexpr], [inherited]
```

**12.275.3.2 alignment**

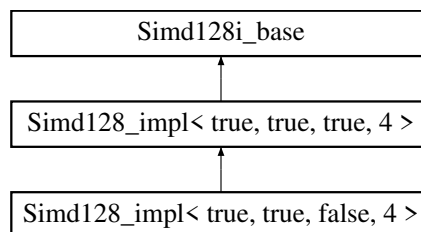
```
const constexpr size_t alignment = 16 [static], [constexpr], [inherited]
```

The documentation for this struct was generated from the following file:

- [simd128\\_int16.inl](#)

**12.276 Simd128\_impl< true, true, false, 4 > Struct Reference**

Inheritance diagram for Simd128\_impl< true, true, false, 4 >:

**Data Structures**

- union [Converter](#)

**Public Types**

- using scalar\_t = uint32\_t
- using aligned\_allocator = [AlignedAllocator](#)< scalar\_t, [Alignment](#)(alignment)>
- using aligned\_vector = std::vector< scalar\_t, aligned\_allocator >
- template<class Field >
  - using is\_same\_element = std::is\_same< typename Field::Element, scalar\_t >
- using vect\_t = \_\_m128i

## Static Public Member Functions

- [static const std::string type\\_string \(\)](#)
- [static INLINE CONST vect\\_t set1 \(const scalar\\_t x\)](#)
- [static INLINE CONST vect\\_t set \(const scalar\\_t x0, const scalar\\_t x1, const scalar\\_t x2, const scalar\\_t x3\)](#)
- [template<class T >](#)  
[static INLINE PURE vect\\_t gather \(const scalar\\_t \\*const p, const T \\*const idx\)](#)
- [static INLINE PURE vect\\_t load \(const scalar\\_t \\*const p\)](#)
- [static INLINE PURE vect\\_t loadu \(const scalar\\_t \\*const p\)](#)
- [static INLINE void store \(scalar\\_t \\*p, vect\\_t v\)](#)
- [static INLINE void storeu \(scalar\\_t \\*p, vect\\_t v\)](#)
- [static INLINE void stream \(scalar\\_t \\*p, const vect\\_t v\)](#)
- [template<int s>](#)  
[static INLINE CONST vect\\_t sra \(const vect\\_t a\)](#)
- [static INLINE CONST vect\\_t greater \(vect\\_t a, vect\\_t b\)](#)
- [static INLINE CONST vect\\_t lesser \(vect\\_t a, vect\\_t b\)](#)
- [static INLINE CONST vect\\_t greater\\_eq \(const vect\\_t a, const vect\\_t b\)](#)
- [static INLINE CONST vect\\_t lesser\\_eq \(const vect\\_t a, const vect\\_t b\)](#)
- [static INLINE CONST vect\\_t mulhi \(const vect\\_t a, const vect\\_t b\)](#)
- [static INLINE CONST vect\\_t mulx \(const vect\\_t a, const vect\\_t b\)](#)
- [static INLINE CONST vect\\_t fmadddx \(const vect\\_t c, const vect\\_t a, const vect\\_t b\)](#)
- [static INLINE vect\\_t fmadddxin \(vect\\_t &c, const vect\\_t a, const vect\\_t b\)](#)
- [static INLINE CONST vect\\_t fnmadddx \(const vect\\_t c, const vect\\_t a, const vect\\_t b\)](#)
- [static INLINE vect\\_t fnmadddxin \(vect\\_t &c, const vect\\_t a, const vect\\_t b\)](#)
- [static INLINE CONST vect\\_t fmsubx \(const vect\\_t c, const vect\\_t a, const vect\\_t b\)](#)
- [static INLINE vect\\_t fmsubxin \(vect\\_t &c, const vect\\_t a, const vect\\_t b\)](#)
- [static INLINE CONST scalar\\_t hadd\\_to\\_scal \(const vect\\_t a\)](#)
- [template<class T >](#)  
[static constexpr bool valid \(T \\*p\)](#)
- [template<class T >](#)  
[static constexpr bool compliant \(T n\)](#)
- [static INLINE CONST vect\\_t set1 \(const scalar\\_t x\)](#)
- [static INLINE CONST vect\\_t set \(const scalar\\_t x0, const scalar\\_t x1, const scalar\\_t x2, const scalar\\_t x3\)](#)
- [template<class T >](#)  
[static INLINE PURE vect\\_t gather \(const scalar\\_t \\*const p, const T \\*const idx\)](#)
- [static INLINE PURE vect\\_t load \(const scalar\\_t \\*const p\)](#)
- [static INLINE PURE vect\\_t loadu \(const scalar\\_t \\*const p\)](#)
- [static INLINE void store \(scalar\\_t \\*p, vect\\_t v\)](#)
- [static INLINE void storeu \(scalar\\_t \\*p, vect\\_t v\)](#)
- [static INLINE void stream \(scalar\\_t \\*p, const vect\\_t v\)](#)
- [template<int s>](#)  
[static INLINE CONST vect\\_t sll \(const vect\\_t a\)](#)
- [template<int s>](#)  
[static INLINE CONST vect\\_t srl \(const vect\\_t a\)](#)
- [template<uint8\\_t s>](#)  
[static INLINE CONST vect\\_t shuffle \(const vect\\_t a\)](#)
- [static INLINE CONST vect\\_t unpacklo\\_intrinsic \(const vect\\_t a, const vect\\_t b\)](#)
- [static INLINE CONST vect\\_t unpackhi\\_intrinsic \(const vect\\_t a, const vect\\_t b\)](#)
- [static INLINE CONST vect\\_t unpacklo \(const vect\\_t a, const vect\\_t b\)](#)
- [static INLINE CONST vect\\_t unpackhi \(const vect\\_t a, const vect\\_t b\)](#)
- [static INLINE void unpacklohi \(vect\\_t &lo, vect\\_t &hi, const vect\\_t a, const vect\\_t b\)](#)
- [static INLINE CONST vect\\_t pack\\_even \(const vect\\_t a, const vect\\_t b\)](#)
- [static INLINE CONST vect\\_t pack\\_odd \(const vect\\_t a, const vect\\_t b\)](#)
- [static INLINE void pack \(vect\\_t &even, vect\\_t &odd, const vect\\_t a, const vect\\_t b\)](#)
- [static INLINE void transpose \(vect\\_t &r0, vect\\_t &r1, vect\\_t &r2, vect\\_t &r3\)](#)

- `template<uint8_t s>`  
`static INLINE CONST vect_t blend (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t add (const vect_t a, const vect_t b)`
- `static INLINE vect_t addin (vect_t &a, const vect_t b)`
- `static INLINE CONST vect_t sub (const vect_t a, const vect_t b)`
- `static INLINE vect_t subin (vect_t &a, const vect_t b)`
- `static INLINE CONST vect_t mullo (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t mul (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmadd (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmadin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fnmadd (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fnmadin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmsub (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmsubin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t eq (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t round (const vect_t a)`
- `static INLINE vect_t mod (vect_t &C, const vect_t &P, const vect_t &INVP, const vect_t &NEGP, const vect_t &MIN, const vect_t &MAX, vect_t &Q, vect_t &T)`
- `static INLINE CONST vect_t zero ()`
- `template<uint8_t s>`  
`static INLINE CONST vect_t sll128 (const vect_t a)`
- `template<uint8_t s>`  
`static INLINE CONST vect_t srl128 (const vect_t a)`
- `static INLINE CONST vect_t vand (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t vor (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t vxor (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t vandnot (const vect_t a, const vect_t b)`

### Static Public Attributes

- `static const constexpr size_t vect_size = 4`
- `static const constexpr size_t alignment = 16`

## 12.276.1 Member Typedef Documentation

### 12.276.1.1 scalar\_t

```
using scalar_t = uint32_t
```

### 12.276.1.2 aligned\_allocator

```
using aligned_allocator = AlignedAllocator<scalar_t, Alignment(alignment)>
```

### 12.276.1.3 aligned\_vector

```
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```

### 12.276.1.4 is\_same\_element

```
template<class Field >
using is_same_element = std::is_same<typename Field::Element, scalar_t>
```

### 12.276.1.5 vect\_t

```
using vect_t = __m128i [inherited]
```

## 12.276.2 Member Function Documentation

### 12.276.2.1 type\_string()

```
static const std::string type_string () [inline], [static]
```

### 12.276.2.2 set1() [1/2]

```
static INLINE CONST vect_t set1 (
 const scalar_t x) [inline], [static]
```

### 12.276.2.3 set() [1/2]

```
static INLINE CONST vect_t set (
 const scalar_t x0,
 const scalar_t x1,
 const scalar_t x2,
 const scalar_t x3) [inline], [static]
```

### 12.276.2.4 gather() [1/2]

```
template<class T >
static INLINE PURE vect_t gather (
 const scalar_t *const p,
 const T *const idx) [inline], [static]
```

### 12.276.2.5 load() [1/2]

```
static INLINE PURE vect_t load (
 const scalar_t *const p) [inline], [static]
```

### 12.276.2.6 loadu() [1/2]

```
static INLINE PURE vect_t loadu (
 const scalar_t *const p) [inline], [static]
```

### 12.276.2.7 store() [1/2]

```
static INLINE void store (
 scalar_t * p,
 vect_t v) [inline], [static]
```

### 12.276.2.8 storeu() [1/2]

```
static INLINE void storeu (
 scalar_t * p,
 vect_t v) [inline], [static]
```

### 12.276.2.9 stream() [1/2]

```
static INLINE void stream (
 scalar_t * p,
 const vect_t v) [inline], [static]
```

### 12.276.2.10 sra()

```
template<int s>
static INLINE CONST vect_t sra (
 const vect_t a) [inline], [static]
```

**12.276.2.11 greater()**

```
static INLINE CONST vect_t greater (
 vect_t a,
 vect_t b) [inline], [static]
```

**12.276.2.12 lesser()**

```
static INLINE CONST vect_t lesser (
 vect_t a,
 vect_t b) [inline], [static]
```

**12.276.2.13 greater\_eq()**

```
static INLINE CONST vect_t greater_eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.276.2.14 lesser\_eq()**

```
static INLINE CONST vect_t lesser_eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.276.2.15 mulhi()**

```
static INLINE CONST vect_t mulhi (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.276.2.16 mulx()**

```
static INLINE CONST vect_t mulx (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.276.2.17 fmaddx()**

```
static INLINE CONST vect_t fmaddx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.276.2.18 fmaddxin()**

```
static INLINE vect_t fmaddxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.276.2.19 fnmaddx()**

```
static INLINE CONST vect_t fnmaddx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.276.2.20 fnmaddxin()**

```
static INLINE vect_t fnmaddxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.276.2.21 fmsubx()**

```
static INLINE CONST vect_t fmsubx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.276.2.22 fmsubxin()**

```
static INLINE vect_t fmsubxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.276.2.23 hadd\_to\_scal()**

```
static INLINE CONST scalar_t hadd_to_scal (
 const vect_t a) [inline], [static]
```

**12.276.2.24 valid()**

```
template<class T >
static constexpr bool valid (
 T * p) [inline], [static], [constexpr], [inherited]
```

**12.276.2.25 compliant()**

```
template<class T >
static constexpr bool compliant (
 T n) [inline], [static], [constexpr], [inherited]
```

**12.276.2.26 set1() [2/2]**

```
static INLINE CONST vect_t set1 (
 const scalar_t x) [inline], [static], [inherited]
```

**12.276.2.27 set() [2/2]**

```
static INLINE CONST vect_t set (
 const scalar_t x0,
 const scalar_t x1,
 const scalar_t x2,
 const scalar_t x3) [inline], [static], [inherited]
```

**12.276.2.28 gather() [2/2]**

```
template<class T >
static INLINE PURE vect_t gather (
 const scalar_t *const p,
 const T *const idx) [inline], [static], [inherited]
```

**12.276.2.29 load()** [2/2]

```
static INLINE PURE vect_t load (
 const scalar_t *const p) [inline], [static], [inherited]
```

**12.276.2.30 loadu()** [2/2]

```
static INLINE PURE vect_t loadu (
 const scalar_t *const p) [inline], [static], [inherited]
```

**12.276.2.31 store()** [2/2]

```
static INLINE void store (
 scalar_t * p,
 vect_t v) [inline], [static], [inherited]
```

**12.276.2.32 storeu()** [2/2]

```
static INLINE void storeu (
 scalar_t * p,
 vect_t v) [inline], [static], [inherited]
```

**12.276.2.33 stream()** [2/2]

```
static INLINE void stream (
 scalar_t * p,
 const vect_t v) [inline], [static], [inherited]
```

**12.276.2.34 sll()**

```
template<int s>
static INLINE CONST vect_t sll (
 const vect_t a) [inline], [static], [inherited]
```

**12.276.2.35 srl()**

```
template<int s>
static INLINE CONST vect_t srl (
 const vect_t a) [inline], [static], [inherited]
```

**12.276.2.36 shuffle()**

```
template<uint8_t s>
static INLINE CONST vect_t shuffle (
 const vect_t a) [inline], [static], [inherited]
```

**12.276.2.37 unpacklo\_intrinsic()**

```
static INLINE CONST vect_t unpacklo_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.276.2.38 unpackhi\_intrinsic()**

```
static INLINE CONST vect_t unpackhi_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```



**12.276.2.39 unpacklo()**

```
static INLINE CONST vect_t unpacklo (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.276.2.40 unpackhi()**

```
static INLINE CONST vect_t unpackhi (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.276.2.41 unpacklohi()**

```
static INLINE void unpacklohi (
 vect_t & lo,
 vect_t & hi,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.276.2.42 pack\_even()**

```
static INLINE CONST vect_t pack_even (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.276.2.43 pack\_odd()**

```
static INLINE CONST vect_t pack_odd (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.276.2.44 pack()**

```
static INLINE void pack (
 vect_t & even,
 vect_t & odd,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.276.2.45 transpose()**

```
static INLINE void transpose (
 vect_t & r0,
 vect_t & r1,
 vect_t & r2,
 vect_t & r3) [inline], [static], [inherited]
```

**12.276.2.46 blend()**

```
template<uint8_t s>
static INLINE CONST vect_t blend (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.276.2.47 add()**

```
static INLINE CONST vect_t add (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.276.2.48 addin()**

```
static INLINE vect_t addin (
 vect_t & a,
 const vect_t b) [inline], [static], [inherited]
```

**12.276.2.49 sub()**

```
static INLINE CONST vect_t sub (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.276.2.50 subin()**

```
static INLINE vect_t subin (
 vect_t & a,
 const vect_t b) [inline], [static], [inherited]
```

**12.276.2.51 mullo()**

```
static INLINE CONST vect_t mullo (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.276.2.52 mul()**

```
static INLINE CONST vect_t mul (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.276.2.53 fmadd()**

```
static INLINE CONST vect_t fmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.276.2.54 fmaddin()**

```
static INLINE vect_t fmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.276.2.55 fnmadd()**

```
static INLINE CONST vect_t fnmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.276.2.56 fnmaddin()**

```
static INLINE vect_t fnmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.276.2.57 fmsub()**

```
static INLINE CONST vect_t fmsub (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.276.2.58 fmsubin()**

```
static INLINE vect_t fmsubin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.276.2.59 eq()**

```
static INLINE CONST vect_t eq (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.276.2.60 round()**

```
static INLINE CONST vect_t round (
 const vect_t a) [inline], [static], [inherited]
```

**12.276.2.61 mod()**

```
static INLINE vect_t mod (
 vect_t & C,
 const vect_t & P,
 const vect_t & INVP,
 const vect_t & NEGP,
 const vect_t & MIN,
 const vect_t & MAX,
 vect_t & Q,
 vect_t & T) [inline], [static], [inherited]
```

**12.276.2.62 zero()**

```
static INLINE CONST vect_t zero () [inline], [static], [inherited]
```

**12.276.2.63 sll128()**

```
template<uint8_t s>
static INLINE CONST vect_t sll128 (
 const vect_t a) [inline], [static], [inherited]
```

**12.276.2.64 srl128()**

```
template<uint8_t s>
static INLINE CONST vect_t srl128 (
 const vect_t a) [inline], [static], [inherited]
```

**12.276.2.65 vand()**

```
static INLINE CONST vect_t vand (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.276.2.66 vor()**

```
static INLINE CONST vect_t vor (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.276.2.67 vxor()**

```
static INLINE CONST vect_t vxor (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.276.2.68 vandnot()**

```
static INLINE CONST vect_t vandnot (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.276.3 Field Documentation****12.276.3.1 vect\_size**

```
const constexpr size_t vect_size = 4 [static], [constexpr], [inherited]
```

**12.276.3.2 alignment**

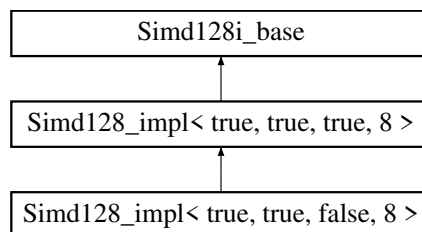
```
const constexpr size_t alignment = 16 [static], [constexpr], [inherited]
```

The documentation for this struct was generated from the following file:

- [simd128\\_int32.inl](#)

**12.277 Simd128\_impl< true, true, false, 8 > Struct Reference**

Inheritance diagram for Simd128\_impl< true, true, false, 8 >:

**Data Structures**

- union [Converter](#)

**Public Types**

- using scalar\_t = uint64\_t
- using aligned\_allocator = [AlignedAllocator](#)< scalar\_t, [Alignment](#)(alignment)>
- using aligned\_vector = std::vector< scalar\_t, aligned\_allocator >
- template<class Field >
  - using is\_same\_element = std::is\_same< typename Field::Element, scalar\_t >
- using vect\_t = \_\_m128i

## Static Public Member Functions

- `static const std::string type_string ()`
- `static INLINE CONST vect_t set1 (const scalar_t x)`
- `static INLINE CONST vect_t set (const scalar_t x0, const scalar_t x1)`
- `template<class T >`  
`static INLINE PURE vect_t gather (const scalar_t *const p, const T *const idx)`
- `static INLINE PURE vect_t load (const scalar_t *const p)`
- `static INLINE PURE vect_t loadu (const scalar_t *const p)`
- `static INLINE void store (scalar_t *p, vect_t v)`
- `static INLINE void storeu (scalar_t *p, vect_t v)`
- `static INLINE void stream (scalar_t *p, const vect_t v)`
- `template<int s>`  
`static INLINE CONST vect_t sra (const vect_t a)`
- `static INLINE CONST vect_t greater (vect_t a, vect_t b)`
- `static INLINE CONST vect_t lesser (vect_t a, vect_t b)`
- `static INLINE CONST vect_t greater_eq (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t lesser_eq (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t mullo (const vect_t x0, const vect_t x1)`
- `static INLINE CONST vect_t mulhi (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t mulx (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmaddx (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmaddxin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fnmaddx (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fnmaddxin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmsubx (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmsubxin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST scalar_t hadd_to_scal (const vect_t a)`
- `template<class T >`  
`static constexpr bool valid (T *p)`
- `template<class T >`  
`static constexpr bool compliant (T n)`
- `static INLINE CONST vect_t set1 (const scalar_t x)`
- `static INLINE CONST vect_t set (const scalar_t x0, const scalar_t x1)`
- `template<class T >`  
`static INLINE PURE vect_t gather (const scalar_t *const p, const T *const idx)`
- `template<int idx>`  
`static INLINE CONST scalar_t get (vect_t v)`
- `static INLINE PURE vect_t load (const scalar_t *const p)`
- `static INLINE PURE vect_t loadu (const scalar_t *const p)`
- `static INLINE void store (scalar_t *p, vect_t v)`
- `static INLINE void storeu (scalar_t *p, vect_t v)`
- `static INLINE void stream (scalar_t *p, const vect_t v)`
- `template<int s>`  
`static INLINE CONST vect_t sll (const vect_t a)`
- `template<int s>`  
`static INLINE CONST vect_t srl (const vect_t a)`
- `template<uint8_t s>`  
`static INLINE CONST vect_t shuffle (const vect_t a)`
- `static INLINE CONST vect_t unpacklo_intrinsic (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t unpackhi_intrinsic (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t unpacklo (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t unpackhi (const vect_t a, const vect_t b)`
- `static INLINE void unpacklohi (vect_t &lo, vect_t &hi, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t pack_even (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t pack_odd (const vect_t a, const vect_t b)`

- `static inline void pack (vect_t &even, vect_t &odd, const vect_t a, const vect_t b)`
- `static inline void transpose (vect_t &r0, vect_t &r1)`
- `template<uint8_t s>`  
`static inline const vect_t blend (const vect_t a, const vect_t b)`
- `static inline const vect_t add (const vect_t a, const vect_t b)`
- `static inline vect_t addin (vect_t &a, const vect_t b)`
- `static inline const vect_t sub (const vect_t a, const vect_t b)`
- `static inline vect_t subin (vect_t &a, const vect_t b)`
- `static inline const vect_t mul (const vect_t a, const vect_t b)`
- `static inline const vect_t fmadd (const vect_t c, const vect_t a, const vect_t b)`
- `static inline vect_t fmaddin (vect_t &c, const vect_t a, const vect_t b)`
- `static inline const vect_t fnmadd (const vect_t c, const vect_t a, const vect_t b)`
- `static inline vect_t fnmaddin (vect_t &c, const vect_t a, const vect_t b)`
- `static inline const vect_t fmsub (const vect_t c, const vect_t a, const vect_t b)`
- `static inline vect_t fmsubin (vect_t &c, const vect_t a, const vect_t b)`
- `static inline const vect_t eq (const vect_t a, const vect_t b)`
- `static inline const vect_t round (const vect_t a)`
- `static inline const vect_t mask_high ()`
- `static inline const vect_t mulhi_fast (vect_t x, vect_t y)`
- `static inline vect_t mod (vect_t &C, const __m128d &P, const __m128d &INVP, const __m128d &NEGP, const vect_t &POW50REM, const __m128d &MIN, const __m128d &MAX, __m128d &Q, __m128d &T)`
- `static inline const vect_t zero ()`
- `template<uint8_t s>`  
`static inline const vect_t sll128 (const vect_t a)`
- `template<uint8_t s>`  
`static inline const vect_t srl128 (const vect_t a)`
- `static inline const vect_t vand (const vect_t a, const vect_t b)`
- `static inline const vect_t vor (const vect_t a, const vect_t b)`
- `static inline const vect_t vxor (const vect_t a, const vect_t b)`
- `static inline const vect_t vandnot (const vect_t a, const vect_t b)`

### Static Public Attributes

- `static const constexpr size_t vect_size = 2`
- `static const constexpr size_t alignment = 16`

### Static Protected Member Functions

- `static inline const vect_t signbits (const vect_t x)`

## 12.277.1 Member Typedef Documentation

### 12.277.1.1 scalar\_t

```
using scalar_t = uint64_t
```

### 12.277.1.2 aligned\_allocator

```
using aligned_allocator = AlignedAllocator<scalar_t, Alignment(alignment)>
```

### 12.277.1.3 aligned\_vector

```
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```

### 12.277.1.4 is\_same\_element

```
template<class Field >
using is_same_element = std::is_same<typename Field::Element, scalar_t>
```

### 12.277.1.5 vect\_t

```
using vect_t = __m128i [inherited]
```

## 12.277.2 Member Function Documentation

### 12.277.2.1 type\_string()

```
static const std::string type_string () [inline], [static]
```

### 12.277.2.2 set1() [1/2]

```
static INLINE CONST vect_t set1 (
 const scalar_t x) [inline], [static]
```

### 12.277.2.3 set() [1/2]

```
static INLINE CONST vect_t set (
 const scalar_t x0,
 const scalar_t x1) [inline], [static]
```

### 12.277.2.4 gather() [1/2]

```
template<class T >
static INLINE PURE vect_t gather (
 const scalar_t *const p,
 const T *const idx) [inline], [static]
```

### 12.277.2.5 load() [1/2]

```
static INLINE PURE vect_t load (
 const scalar_t *const p) [inline], [static]
```

### 12.277.2.6 loadu() [1/2]

```
static INLINE PURE vect_t loadu (
 const scalar_t *const p) [inline], [static]
```

### 12.277.2.7 store() [1/2]

```
static INLINE void store (
 scalar_t * p,
 vect_t v) [inline], [static]
```

### 12.277.2.8 storeu() [1/2]

```
static INLINE void storeu (
 scalar_t * p,
 vect_t v) [inline], [static]
```

### 12.277.2.9 stream() [1/2]

```
static INLINE void stream (
 scalar_t * p,
 const vect_t v) [inline], [static]
```

### 12.277.2.10 sra()

```
template<int s>
static INLINE CONST vect_t sra (
 const vect_t a) [inline], [static]
```

**12.277.2.11 greater()**

```
static INLINE CONST vect_t greater (
 vect_t a,
 vect_t b) [inline], [static]
```

**12.277.2.12 lesser()**

```
static INLINE CONST vect_t lesser (
 vect_t a,
 vect_t b) [inline], [static]
```

**12.277.2.13 greater\_eq()**

```
static INLINE CONST vect_t greater_eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.277.2.14 lesser\_eq()**

```
static INLINE CONST vect_t lesser_eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.277.2.15 mullo()**

```
static INLINE CONST vect_t mullo (
 const vect_t x0,
 const vect_t x1) [inline], [static]
```

**12.277.2.16 mulhi()**

```
static INLINE CONST vect_t mulhi (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.277.2.17 mulx()**

```
static INLINE CONST vect_t mulx (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.277.2.18 fmaddx()**

```
static INLINE CONST vect_t fmaddx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.277.2.19 fmaddxin()**

```
static INLINE vect_t fmaddxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```



**12.277.2.20 fnmaddx()**

```
static INLINE CONST vect_t fnmaddx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.277.2.21 fnmaddxin()**

```
static INLINE vect_t fnmaddxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.277.2.22 fmsubx()**

```
static INLINE CONST vect_t fmsubx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.277.2.23 fmsubxin()**

```
static INLINE vect_t fmsubxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.277.2.24 hadd\_to\_scal()**

```
static INLINE CONST scalar_t hadd_to_scal (
 const vect_t a) [inline], [static]
```

**12.277.2.25 valid()**

```
template<class T >
static constexpr bool valid (
 T * p) [inline], [static], [constexpr], [inherited]
```

**12.277.2.26 compliant()**

```
template<class T >
static constexpr bool compliant (
 T n) [inline], [static], [constexpr], [inherited]
```

**12.277.2.27 set1() [2/2]**

```
static INLINE CONST vect_t set1 (
 const scalar_t x) [inline], [static], [inherited]
```

**12.277.2.28 set() [2/2]**

```
static INLINE CONST vect_t set (
 const scalar_t x0,
 const scalar_t x1) [inline], [static], [inherited]
```

**12.277.2.29 gather()** [2/2]

```
template<class T >
static INLINE PURE vect_t gather (
 const scalar_t *const p,
 const T *const idx) [inline], [static], [inherited]
```

**12.277.2.30 get()**

```
template<int idx>
static INLINE CONST scalar_t get (
 vect_t v) [inline], [static], [inherited]
```

**12.277.2.31 load()** [2/2]

```
static INLINE PURE vect_t load (
 const scalar_t *const p) [inline], [static], [inherited]
```

**12.277.2.32 loadu()** [2/2]

```
static INLINE PURE vect_t loadu (
 const scalar_t *const p) [inline], [static], [inherited]
```

**12.277.2.33 store()** [2/2]

```
static INLINE void store (
 scalar_t * p,
 vect_t v) [inline], [static], [inherited]
```

**12.277.2.34 storeu()** [2/2]

```
static INLINE void storeu (
 scalar_t * p,
 vect_t v) [inline], [static], [inherited]
```

**12.277.2.35 stream()** [2/2]

```
static INLINE void stream (
 scalar_t * p,
 const vect_t v) [inline], [static], [inherited]
```

**12.277.2.36 sll()**

```
template<int s>
static INLINE CONST vect_t sll (
 const vect_t a) [inline], [static], [inherited]
```

**12.277.2.37 srl()**

```
template<int s>
static INLINE CONST vect_t srl (
 const vect_t a) [inline], [static], [inherited]
```

**12.277.2.38 shuffle()**

```
template<uint8_t s>
static INLINE CONST vect_t shuffle (
 const vect_t a) [inline], [static], [inherited]
```

**12.277.2.39 unpacklo\_intrinsic()**

```
static INLINE CONST vect_t unpacklo_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.277.2.40 unpackhi\_intrinsic()**

```
static INLINE CONST vect_t unpackhi_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.277.2.41 unpacklo()**

```
static INLINE CONST vect_t unpacklo (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.277.2.42 unpackhi()**

```
static INLINE CONST vect_t unpackhi (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.277.2.43 unpacklohi()**

```
static INLINE void unpacklohi (
 vect_t & lo,
 vect_t & hi,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.277.2.44 pack\_even()**

```
static INLINE CONST vect_t pack_even (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.277.2.45 pack\_odd()**

```
static INLINE CONST vect_t pack_odd (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.277.2.46 pack()**

```
static INLINE void pack (
 vect_t & even,
 vect_t & odd,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.277.2.47 transpose()**

```
static INLINE void transpose (
 vect_t & r0,
 vect_t & r1) [inline], [static], [inherited]
```

**12.277.2.48 blend()**

```
template<uint8_t s>
static INLINE CONST vect_t blend (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.277.2.49 add()**

```
static INLINE CONST vect_t add (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.277.2.50 addin()**

```
static INLINE vect_t addin (
 vect_t & a,
 const vect_t b) [inline], [static], [inherited]
```

**12.277.2.51 sub()**

```
static INLINE CONST vect_t sub (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.277.2.52 subin()**

```
static INLINE vect_t subin (
 vect_t & a,
 const vect_t b) [inline], [static], [inherited]
```

**12.277.2.53 mul()**

```
static INLINE CONST vect_t mul (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.277.2.54 fmadd()**

```
static INLINE CONST vect_t fmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.277.2.55 fmaddin()**

```
static INLINE vect_t fmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.277.2.56 fnmadd()**

```
static INLINE CONST vect_t fnmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.277.2.57 fnmaddin()**

```
static INLINE vect_t fnmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.277.2.58 fmsub()**

```
static INLINE CONST vect_t fmsub (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.277.2.59 fmsubin()**

```
static INLINE vect_t fmsubin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.277.2.60 eq()**

```
static INLINE CONST vect_t eq (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.277.2.61 round()**

```
static INLINE CONST vect_t round (
 const vect_t a) [inline], [static], [inherited]
```

**12.277.2.62 mask\_high()**

```
static INLINE CONST vect_t mask_high () [inline], [static], [inherited]
```

**12.277.2.63 mulhi\_fast()**

```
INLINE CONST vect_t mulhi_fast (
 vect_t x,
 vect_t y) [static], [inherited]
```

**12.277.2.64 mod()**

```
INLINE vect_t mod (
 vect_t & C,
 const __m128d & P,
 const __m128d & INVP,
 const __m128d & NEGP,
 const vect_t & POW50REM,
 const __m128d & MIN,
 const __m128d & MAX,
 __m128d & Q,
 __m128d & T) [static], [inherited]
```

**12.277.2.65 signbits()**

```
static INLINE CONST vect_t signbits (
 const vect_t x) [inline], [static], [protected], [inherited]
```

**12.277.2.66 zero()**

```
static INLINE CONST vect_t zero () [inline], [static], [inherited]
```

**12.277.2.67 sll128()**

```
template<uint8_t s>
static INLINE CONST vect_t sll128 (
 const vect_t a) [inline], [static], [inherited]
```

**12.277.2.68 srl128()**

```
template<uint8_t s>
static INLINE CONST vect_t srl128 (
 const vect_t a) [inline], [static], [inherited]
```

**12.277.2.69 vand()**

```
static INLINE CONST vect_t vand (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.277.2.70 vor()**

```
static INLINE CONST vect_t vor (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.277.2.71 vxor()**

```
static INLINE CONST vect_t vxor (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.277.2.72 vandnot()**

```
static INLINE CONST vect_t vandnot (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.277.3 Field Documentation****12.277.3.1 vect\_size**

```
const constexpr size_t vect_size = 2 [static], [constexpr], [inherited]
```

**12.277.3.2 alignment**

```
const constexpr size_t alignment = 16 [static], [constexpr], [inherited]
```

The documentation for this struct was generated from the following file:

- [simd128\\_int64.inl](#)

**12.278 Simd128\_impl< true, true, true, 2 > Struct Reference**

Inheritance diagram for Simd128\_impl< true, true, true, 2 >:



## Data Structures

- union [Converter](#)

## Public Types

- using [vect\\_t](#) = [\\_\\_m128i](#)
- using [scalar\\_t](#) = [int16\\_t](#)
- using [aligned\\_allocator](#) = [AlignedAllocator](#)< [scalar\\_t](#), [Alignment](#)([alignment](#))>
- using [aligned\\_vector](#) = [std::vector](#)< [scalar\\_t](#), [aligned\\_allocator](#) >
- template<class [Field](#) >  
using [is\\_same\\_element](#) = [std::is\\_same](#)< [typename Field::Element](#), [scalar\\_t](#) >

## Static Public Member Functions

- static const [std::string](#) [type\\_string](#) ()
- template<class [T](#) >  
static constexpr bool [valid](#) ([T](#) \*[p](#))
- template<class [T](#) >  
static constexpr bool [compliant](#) ([T](#) [n](#))
- static INLINE CONST [vect\\_t](#) [set1](#) ([const scalar\\_t](#) [x](#))
- static INLINE CONST [vect\\_t](#) [set](#) ([const scalar\\_t](#) [x0](#), [const scalar\\_t](#) [x1](#), [const scalar\\_t](#) [x2](#), [const scalar\\_t](#) [x3](#),  
[const scalar\\_t](#) [x4](#), [const scalar\\_t](#) [x5](#), [const scalar\\_t](#) [x6](#), [const scalar\\_t](#) [x7](#))
- template<class [T](#) >  
static INLINE PURE [vect\\_t](#) [gather](#) ([const scalar\\_t](#) \*[const p](#), [const T](#) \*[const idx](#))
- static INLINE PURE [vect\\_t](#) [load](#) ([const scalar\\_t](#) \*[const p](#))
- static INLINE PURE [vect\\_t](#) [loadu](#) ([const scalar\\_t](#) \*[const p](#))
- static INLINE void [store](#) ([scalar\\_t](#) \*[p](#), [vect\\_t](#) [v](#))
- static INLINE void [storeu](#) ([scalar\\_t](#) \*[p](#), [vect\\_t](#) [v](#))
- static INLINE void [stream](#) ([scalar\\_t](#) \*[p](#), [const vect\\_t](#) [v](#))
- template<int [s](#)>  
static INLINE CONST [vect\\_t](#) [sll](#) ([const vect\\_t](#) [a](#))
- template<int [s](#)>  
static INLINE CONST [vect\\_t](#) [srl](#) ([const vect\\_t](#) [a](#))
- template<int [s](#)>  
static INLINE CONST [vect\\_t](#) [sra](#) ([const vect\\_t](#) [a](#))
- template<uint32\_t [s](#)>  
static INLINE CONST [vect\\_t](#) [shuffle](#) ([const vect\\_t](#) [a](#))
- static INLINE CONST [vect\\_t](#) [unpacklo\\_intrinsic](#) ([const vect\\_t](#) [a](#), [const vect\\_t](#) [b](#))
- static INLINE CONST [vect\\_t](#) [unpackhi\\_intrinsic](#) ([const vect\\_t](#) [a](#), [const vect\\_t](#) [b](#))
- static INLINE CONST [vect\\_t](#) [unpacklo](#) ([const vect\\_t](#) [a](#), [const vect\\_t](#) [b](#))
- static INLINE CONST [vect\\_t](#) [unpackhi](#) ([const vect\\_t](#) [a](#), [const vect\\_t](#) [b](#))
- static INLINE void [unpacklohi](#) ([vect\\_t](#) &[lo](#), [vect\\_t](#) &[hi](#), [const vect\\_t](#) [a](#), [const vect\\_t](#) [b](#))
- static INLINE CONST [vect\\_t](#) [pack\\_even](#) ([const vect\\_t](#) [a](#), [const vect\\_t](#) [b](#))
- static INLINE CONST [vect\\_t](#) [pack\\_odd](#) ([const vect\\_t](#) [a](#), [const vect\\_t](#) [b](#))
- static INLINE void [pack](#) ([vect\\_t](#) &[even](#), [vect\\_t](#) &[odd](#), [const vect\\_t](#) [a](#), [const vect\\_t](#) [b](#))
- static INLINE void [transpose](#) ([vect\\_t](#) &[r0](#), [vect\\_t](#) &[r1](#), [vect\\_t](#) &[r2](#), [vect\\_t](#) &[r3](#), [vect\\_t](#) &[r4](#), [vect\\_t](#) &[r5](#), [vect\\_t](#) &[r6](#),  
[vect\\_t](#) &[r7](#))

- `template<uint8_t s>`  
`static INLINE CONST vect_t blend (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t add (const vect_t a, const vect_t b)`
- `static INLINE vect_t addin (vect_t &a, const vect_t b)`
- `static INLINE CONST vect_t sub (const vect_t a, const vect_t b)`
- `static INLINE vect_t subin (vect_t &a, const vect_t b)`
- `static INLINE CONST vect_t mullo (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t mul (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t mulhi (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t mulx (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmaddd (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmadddin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmadddx (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmadddxin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fnmadd (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fnmaddin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fnmaddx (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fnmaddxin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmsub (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmsubin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmsubx (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmsubxin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t eq (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t greater (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t lesser (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t greater_eq (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t lesser_eq (const vect_t a, const vect_t b)`
- `static INLINE CONST scalar_t hadd_to_scal (const vect_t a)`
- `static INLINE CONST vect_t round (const vect_t a)`
- `static INLINE vect_t mod (vect_t &C, const vect_t &P, const __m64 &INVP, const vect_t &NEGP, const vect_t &MIN, const vect_t &MAX, vect_t &Q, vect_t &T)`
- `static INLINE CONST vect_t zero ()`
- `template<uint8_t s>`  
`static INLINE CONST vect_t sll128 (const vect_t a)`
- `template<uint8_t s>`  
`static INLINE CONST vect_t srl128 (const vect_t a)`
- `static INLINE CONST vect_t vand (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t vor (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t vxor (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t vandnot (const vect_t a, const vect_t b)`

### Static Public Attributes

- `static const constexpr size_t vect_size = 8`
- `static const constexpr size_t alignment = 16`

## 12.278.1 Member Typedef Documentation

### 12.278.1.1 vect\_t

```
using vect_t = __m128i
```

### 12.278.1.2 scalar\_t

```
using scalar_t = int16_t
```



**12.278.1.3 aligned\_allocator**

```
using aligned_allocator = AlignedAllocator<scalar_t, Alignment(alignment)>
```

**12.278.1.4 aligned\_vector**

```
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```

**12.278.1.5 is\_same\_element**

```
template<class Field >
using is_same_element = std::is_same<typename Field::Element, scalar_t>
```

**12.278.2 Member Function Documentation****12.278.2.1 type\_string()**

```
static const std::string type_string () [inline], [static]
```

**12.278.2.2 valid()**

```
template<class T >
static constexpr bool valid (
 T * p) [inline], [static], [constexpr]
```

**12.278.2.3 compliant()**

```
template<class T >
static constexpr bool compliant (
 T n) [inline], [static], [constexpr]
```

**12.278.2.4 set1()**

```
static INLINE CONST vect_t set1 (
 const scalar_t x) [inline], [static]
```

**12.278.2.5 set()**

```
static INLINE CONST vect_t set (
 const scalar_t x0,
 const scalar_t x1,
 const scalar_t x2,
 const scalar_t x3,
 const scalar_t x4,
 const scalar_t x5,
 const scalar_t x6,
 const scalar_t x7) [inline], [static]
```

**12.278.2.6 gather()**

```
template<class T >
static INLINE PURE vect_t gather (
 const scalar_t *const p,
 const T *const idx) [inline], [static]
```

**12.278.2.7 load()**

```
static INLINE PURE vect_t load (
 const scalar_t *const p) [inline], [static]
```

**12.278.2.8 loadu()**

```
static INLINE PURE vect_t loadu (
 const scalar_t *const p) [inline], [static]
```

**12.278.2.9 store()**

```
static INLINE void store (
 scalar_t * p,
 vect_t v) [inline], [static]
```

**12.278.2.10 storeu()**

```
static INLINE void storeu (
 scalar_t * p,
 vect_t v) [inline], [static]
```

**12.278.2.11 stream()**

```
static INLINE void stream (
 scalar_t * p,
 const vect_t v) [inline], [static]
```

**12.278.2.12 sll()**

```
template<int s>
static INLINE CONST vect_t sll (
 const vect_t a) [inline], [static]
```

**12.278.2.13 srl()**

```
template<int s>
static INLINE CONST vect_t srl (
 const vect_t a) [inline], [static]
```

**12.278.2.14 sra()**

```
template<int s>
static INLINE CONST vect_t sra (
 const vect_t a) [inline], [static]
```

**12.278.2.15 shuffle()**

```
template<uint32_t s>
static INLINE CONST vect_t shuffle (
 const vect_t a) [inline], [static]
```

**12.278.2.16 unpacklo\_intrinsic()**

```
static INLINE CONST vect_t unpacklo_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.278.2.17 unpackhi\_intrinsic()**

```
static INLINE CONST vect_t unpackhi_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.278.2.18 unpacklo()**

```
static INLINE CONST vect_t unpacklo (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.278.2.19 unpackhi()**

```
static INLINE CONST vect_t unpackhi (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.278.2.20 unpacklohi()**

```
static INLINE void unpacklohi (
 vect_t & lo,
 vect_t & hi,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.278.2.21 pack\_even()**

```
static INLINE CONST vect_t pack_even (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.278.2.22 pack\_odd()**

```
static INLINE CONST vect_t pack_odd (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.278.2.23 pack()**

```
static INLINE void pack (
 vect_t & even,
 vect_t & odd,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.278.2.24 transpose()**

```
static INLINE void transpose (
 vect_t & r0,
 vect_t & r1,
 vect_t & r2,
 vect_t & r3,
 vect_t & r4,
 vect_t & r5,
 vect_t & r6,
 vect_t & r7) [inline], [static]
```

**12.278.2.25 blend()**

```
template<uint8_t s>
static INLINE CONST vect_t blend (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.278.2.26 add()**

```
static INLINE CONST vect_t add (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.278.2.27 addin()**

```
static INLINE vect_t addin (
 vect_t & a,
 const vect_t b) [inline], [static]
```

**12.278.2.28 sub()**

```
static INLINE CONST vect_t sub (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.278.2.29 subin()**

```
static INLINE vect_t subin (
 vect_t & a,
 const vect_t b) [inline], [static]
```

**12.278.2.30 mullo()**

```
static INLINE CONST vect_t mullo (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.278.2.31 mul()**

```
static INLINE CONST vect_t mul (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.278.2.32 mulhi()**

```
static INLINE CONST vect_t mulhi (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.278.2.33 mulx()**

```
static INLINE CONST vect_t mulx (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.278.2.34 fmadd()**

```
static INLINE CONST vect_t fmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.278.2.35 fmaddin()**

```
static INLINE vect_t fmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.278.2.36 fmaddx()**

```
static INLINE CONST vect_t fmaddx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.278.2.37 fmaddxin()**

```
static INLINE vect_t fmaddxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.278.2.38 fnmadd()**

```
static INLINE CONST vect_t fnmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.278.2.39 fnmaddin()**

```
static INLINE vect_t fnmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.278.2.40 fnmaddx()**

```
static INLINE CONST vect_t fnmaddx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.278.2.41 fnmaddxin()**

```
static INLINE vect_t fnmaddxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.278.2.42 fmsub()**

```
static INLINE CONST vect_t fmsub (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.278.2.43 fmsubin()**

```
static INLINE vect_t fmsubin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.278.2.44 fmsubx()**

```
static INLINE CONST vect_t fmsubx (
 const vect_t c,
```

```

 const vect_t a,
 const vect_t b) [inline], [static]

```

#### 12.278.2.45 fmsubxin()

```

static INLINE vect_t fmsubxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]

```

#### 12.278.2.46 eq()

```

static INLINE CONST vect_t eq (
 const vect_t a,
 const vect_t b) [inline], [static]

```

#### 12.278.2.47 greater()

```

static INLINE CONST vect_t greater (
 const vect_t a,
 const vect_t b) [inline], [static]

```

#### 12.278.2.48 lesser()

```

static INLINE CONST vect_t lesser (
 const vect_t a,
 const vect_t b) [inline], [static]

```

#### 12.278.2.49 greater\_eq()

```

static INLINE CONST vect_t greater_eq (
 const vect_t a,
 const vect_t b) [inline], [static]

```

#### 12.278.2.50 lesser\_eq()

```

static INLINE CONST vect_t lesser_eq (
 const vect_t a,
 const vect_t b) [inline], [static]

```

#### 12.278.2.51 hadd\_to\_scal()

```

static INLINE CONST scalar_t hadd_to_scal (
 const vect_t a) [inline], [static]

```

#### 12.278.2.52 round()

```

static INLINE CONST vect_t round (
 const vect_t a) [inline], [static]

```

#### 12.278.2.53 mod()

```

static INLINE vect_t mod (
 vect_t & C,
 const vect_t & P,
 const __m64 & INVP,
 const vect_t & NEGP,
 const vect_t & MIN,
 const vect_t & MAX,

```

```

 vect_t & Q,
 vect_t & T) [inline], [static]

```

#### 12.278.2.54 zero()

```
static INLINE CONST vect_t zero () [inline], [static], [inherited]
```

#### 12.278.2.55 sll128()

```

template<uint8_t s>
static INLINE CONST vect_t sll128 (
 const vect_t a) [inline], [static], [inherited]

```

#### 12.278.2.56 srl128()

```

template<uint8_t s>
static INLINE CONST vect_t srl128 (
 const vect_t a) [inline], [static], [inherited]

```

#### 12.278.2.57 vand()

```

static INLINE CONST vect_t vand (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]

```

#### 12.278.2.58 vor()

```

static INLINE CONST vect_t vor (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]

```

#### 12.278.2.59 vxor()

```

static INLINE CONST vect_t vxor (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]

```

#### 12.278.2.60 vandnot()

```

static INLINE CONST vect_t vandnot (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]

```

### 12.278.3 Field Documentation

#### 12.278.3.1 vect\_size

```
const constexpr size_t vect_size = 8 [static], [constexpr]
```

#### 12.278.3.2 alignment

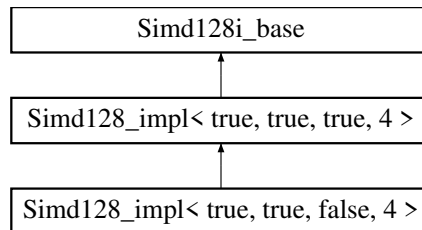
```
const constexpr size_t alignment = 16 [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [simd128\\_int16.inl](#)

## 12.279 Simd128\_impl< true, true, true, 4 > Struct Reference

Inheritance diagram for Simd128\_impl< true, true, true, 4 >:



### Data Structures

- union [Converter](#)

### Public Types

- using vect\_t = \_\_m128i
- using scalar\_t = int32\_t
- using aligned\_allocator = AlignedAllocator< scalar\_t, Alignment(alignment)>
- using aligned\_vector = std::vector< scalar\_t, aligned\_allocator >
- template<class Field >  
using is\_same\_element = std::is\_same< typename Field::Element, scalar\_t >

### Static Public Member Functions

- static const std::string type\_string ()
- template<class T >  
static constexpr bool valid (T \*p)
- template<class T >  
static constexpr bool compliant (T n)
- static INLINE CONST vect\_t set1 (const scalar\_t x)
- static INLINE CONST vect\_t set (const scalar\_t x0, const scalar\_t x1, const scalar\_t x2, const scalar\_t x3)
- template<class T >  
static INLINE PURE vect\_t gather (const scalar\_t \*const p, const T \*const idx)
- static INLINE PURE vect\_t load (const scalar\_t \*const p)
- static INLINE PURE vect\_t loadu (const scalar\_t \*const p)
- static INLINE void store (scalar\_t \*p, vect\_t v)
- static INLINE void storeu (scalar\_t \*p, vect\_t v)
- static INLINE void stream (scalar\_t \*p, const vect\_t v)
- template<int s>  
static INLINE CONST vect\_t sll (const vect\_t a)
- template<int s>  
static INLINE CONST vect\_t srl (const vect\_t a)
- template<int s>  
static INLINE CONST vect\_t sra (const vect\_t a)
- template<uint8\_t s>  
static INLINE CONST vect\_t shuffle (const vect\_t a)
- static INLINE CONST vect\_t unpacklo\_intrinsic (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t unpackhi\_intrinsic (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t unpacklo (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t unpackhi (const vect\_t a, const vect\_t b)
- static INLINE void unpacklohi (vect\_t &lo, vect\_t &hi, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t pack\_even (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t pack\_odd (const vect\_t a, const vect\_t b)



- static INLINE void pack (vect\_t &even, vect\_t &odd, const vect\_t a, const vect\_t b)
- static INLINE void transpose (vect\_t &r0, vect\_t &r1, vect\_t &r2, vect\_t &r3)
- template<uint8\_t s>
  - static INLINE CONST vect\_t blend (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t add (const vect\_t a, const vect\_t b)
- static INLINE vect\_t addin (vect\_t &a, const vect\_t b)
- static INLINE CONST vect\_t sub (const vect\_t a, const vect\_t b)
- static INLINE vect\_t subin (vect\_t &a, const vect\_t b)
- static INLINE CONST vect\_t mullo (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t mul (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t mulhi (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t mulx (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fmadd (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fmadddin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fmadddx (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fmadddxin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fnmadd (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fnmadddin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fnmadddx (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fnmadddxin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fmsub (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fmsubin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fmsubx (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fmsubxin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t eq (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t greater (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t lesser (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t greater\_eq (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t lesser\_eq (const vect\_t a, const vect\_t b)
- static INLINE CONST scalar\_t hadd\_to\_scal (const vect\_t a)
- static INLINE CONST vect\_t round (const vect\_t a)
- static INLINE vect\_t mod (vect\_t &C, const vect\_t &P, const vect\_t &INVP, const vect\_t &NEGP, const vect\_t &MIN, const vect\_t &MAX, vect\_t &Q, vect\_t &T)
- static INLINE CONST vect\_t zero ()
- template<uint8\_t s>
  - static INLINE CONST vect\_t sll128 (const vect\_t a)
- template<uint8\_t s>
  - static INLINE CONST vect\_t srl128 (const vect\_t a)
- static INLINE CONST vect\_t vand (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t vor (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t vxor (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t vandnot (const vect\_t a, const vect\_t b)

### Static Public Attributes

- static const constexpr size\_t vect\_size = 4
- static const constexpr size\_t alignment = 16

## 12.279.1 Member Typedef Documentation

### 12.279.1.1 vect\_t

```
using vect_t = __m128i
```

### 12.279.1.2 scalar\_t

```
using scalar_t = int32_t
```

**12.279.1.3 aligned\_allocator**

```
using aligned_allocator = AlignedAllocator<scalar_t, Alignment(alignment)>
```

**12.279.1.4 aligned\_vector**

```
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```

**12.279.1.5 is\_same\_element**

```
template<class Field >
using is_same_element = std::is_same<typename Field::Element, scalar_t>
```

**12.279.2 Member Function Documentation****12.279.2.1 type\_string()**

```
static const std::string type_string () [inline], [static]
```

**12.279.2.2 valid()**

```
template<class T >
static constexpr bool valid (
 T * p) [inline], [static], [constexpr]
```

**12.279.2.3 compliant()**

```
template<class T >
static constexpr bool compliant (
 T n) [inline], [static], [constexpr]
```

**12.279.2.4 set1()**

```
static INLINE CONST vect_t set1 (
 const scalar_t x) [inline], [static]
```

**12.279.2.5 set()**

```
static INLINE CONST vect_t set (
 const scalar_t x0,
 const scalar_t x1,
 const scalar_t x2,
 const scalar_t x3) [inline], [static]
```

**12.279.2.6 gather()**

```
template<class T >
static INLINE PURE vect_t gather (
 const scalar_t *const p,
 const T *const idx) [inline], [static]
```

**12.279.2.7 load()**

```
static INLINE PURE vect_t load (
 const scalar_t *const p) [inline], [static]
```

**12.279.2.8 loadu()**

```
static INLINE PURE vect_t loadu (
 const scalar_t *const p) [inline], [static]
```

**12.279.2.9 store()**

```
static INLINE void store (
 scalar_t * p,
 vect_t v) [inline], [static]
```

**12.279.2.10 storeu()**

```
static INLINE void storeu (
 scalar_t * p,
 vect_t v) [inline], [static]
```

**12.279.2.11 stream()**

```
static INLINE void stream (
 scalar_t * p,
 const vect_t v) [inline], [static]
```

**12.279.2.12 sll()**

```
template<int s>
static INLINE CONST vect_t sll (
 const vect_t a) [inline], [static]
```

**12.279.2.13 srl()**

```
template<int s>
static INLINE CONST vect_t srl (
 const vect_t a) [inline], [static]
```

**12.279.2.14 sra()**

```
template<int s>
static INLINE CONST vect_t sra (
 const vect_t a) [inline], [static]
```

**12.279.2.15 shuffle()**

```
template<uint8_t s>
static INLINE CONST vect_t shuffle (
 const vect_t a) [inline], [static]
```

**12.279.2.16 unpacklo\_intrinsic()**

```
static INLINE CONST vect_t unpacklo_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.279.2.17 unpackhi\_intrinsic()**

```
static INLINE CONST vect_t unpackhi_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.279.2.18 unpacklo()**

```
static INLINE CONST vect_t unpacklo (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.279.2.19 unpackhi()**

```
static INLINE CONST vect_t unpackhi (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.279.2.20 unpacklohi()**

```
static INLINE void unpacklohi (
 vect_t & lo,
 vect_t & hi,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.279.2.21 pack\_even()**

```
static INLINE CONST vect_t pack_even (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.279.2.22 pack\_odd()**

```
static INLINE CONST vect_t pack_odd (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.279.2.23 pack()**

```
static INLINE void pack (
 vect_t & even,
 vect_t & odd,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.279.2.24 transpose()**

```
static INLINE void transpose (
 vect_t & r0,
 vect_t & r1,
 vect_t & r2,
 vect_t & r3) [inline], [static]
```

**12.279.2.25 blend()**

```
template<uint8_t s>
static INLINE CONST vect_t blend (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.279.2.26 add()**

```
static INLINE CONST vect_t add (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.279.2.27 addin()**

```
static INLINE vect_t addin (
 vect_t & a,
 const vect_t b) [inline], [static]
```

**12.279.2.28 sub()**

```
static INLINE CONST vect_t sub (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.279.2.29 subin()**

```
static INLINE vect_t subin (
 vect_t & a,
 const vect_t b) [inline], [static]
```

**12.279.2.30 mullo()**

```
static INLINE CONST vect_t mullo (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.279.2.31 mul()**

```
static INLINE CONST vect_t mul (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.279.2.32 mulhi()**

```
static INLINE CONST vect_t mulhi (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.279.2.33 mulx()**

```
static INLINE CONST vect_t mulx (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.279.2.34 fmadd()**

```
static INLINE CONST vect_t fmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.279.2.35 fmaddin()**

```
static INLINE vect_t fmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.279.2.36 fmaddx()**

```
static INLINE CONST vect_t fmaddx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.279.2.37 fmaddxin()**

```
static INLINE vect_t fmaddxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.279.2.38 fnmadd()**

```
static INLINE CONST vect_t fnmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.279.2.39 fnmaddin()**

```
static INLINE vect_t fnmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.279.2.40 fnmaddx()**

```
static INLINE CONST vect_t fnmaddx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.279.2.41 fnmaddxin()**

```
static INLINE vect_t fnmaddxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.279.2.42 fmsub()**

```
static INLINE CONST vect_t fmsub (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.279.2.43 fmsubin()**

```
static INLINE vect_t fmsubin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.279.2.44 fmsubx()**

```
static INLINE CONST vect_t fmsubx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.279.2.45 fmsubxin()**

```
static INLINE vect_t fmsubxin (
 vect_t & c,
```

```

 const vect_t a,
 const vect_t b) [inline], [static]

```

**12.279.2.46 eq()**

```

static INLINE CONST vect_t eq (
 const vect_t a,
 const vect_t b) [inline], [static]

```

**12.279.2.47 greater()**

```

static INLINE CONST vect_t greater (
 const vect_t a,
 const vect_t b) [inline], [static]

```

**12.279.2.48 lesser()**

```

static INLINE CONST vect_t lesser (
 const vect_t a,
 const vect_t b) [inline], [static]

```

**12.279.2.49 greater\_eq()**

```

static INLINE CONST vect_t greater_eq (
 const vect_t a,
 const vect_t b) [inline], [static]

```

**12.279.2.50 lesser\_eq()**

```

static INLINE CONST vect_t lesser_eq (
 const vect_t a,
 const vect_t b) [inline], [static]

```

**12.279.2.51 hadd\_to\_scal()**

```

static INLINE CONST scalar_t hadd_to_scal (
 const vect_t a) [inline], [static]

```

**12.279.2.52 round()**

```

static INLINE CONST vect_t round (
 const vect_t a) [inline], [static]

```

**12.279.2.53 mod()**

```

static INLINE vect_t mod (
 vect_t & C,
 const vect_t & P,
 const vect_t & INVP,
 const vect_t & NEGP,
 const vect_t & MIN,
 const vect_t & MAX,
 vect_t & Q,
 vect_t & T) [inline], [static]

```

**12.279.2.54 zero()**

```

static INLINE CONST vect_t zero () [inline], [static], [inherited]

```

**12.279.2.55 sll128()**

```
template<uint8_t s>
static INLINE CONST vect_t sll128 (
 const vect_t a) [inline], [static], [inherited]
```

**12.279.2.56 srl128()**

```
template<uint8_t s>
static INLINE CONST vect_t srl128 (
 const vect_t a) [inline], [static], [inherited]
```

**12.279.2.57 vand()**

```
static INLINE CONST vect_t vand (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.279.2.58 vor()**

```
static INLINE CONST vect_t vor (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.279.2.59 vxor()**

```
static INLINE CONST vect_t vxor (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.279.2.60 vandnot()**

```
static INLINE CONST vect_t vandnot (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.279.3 Field Documentation****12.279.3.1 vect\_size**

```
const constexpr size_t vect_size = 4 [static], [constexpr]
```

**12.279.3.2 alignment**

```
const constexpr size_t alignment = 16 [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [simd128\\_int32.inl](#)

**12.280 Simd128\_impl< true, true, true, 8 > Struct Reference**

Inheritance diagram for Simd128\_impl< true, true, true, 8 >:





## Data Structures

- union [Converter](#)

## Public Types

- using [vect\\_t](#) = [\\_\\_m128i](#)
- using [scalar\\_t](#) = [int64\\_t](#)
- using [aligned\\_allocator](#) = [AlignedAllocator](#)< [scalar\\_t](#), [Alignment](#)([alignment](#))>
- using [aligned\\_vector](#) = [std::vector](#)< [scalar\\_t](#), [aligned\\_allocator](#) >
- template<class [Field](#) >  
using [is\\_same\\_element](#) = [std::is\\_same](#)< [typename Field::Element](#), [scalar\\_t](#) >

## Static Public Member Functions

- static const [std::string](#) [type\\_string](#) ()
- template<class [T](#) >  
static constexpr bool [valid](#) ([T \\*p](#))
- template<class [T](#) >  
static constexpr bool [compliant](#) ([T n](#))
- static INLINE CONST [vect\\_t](#) [set1](#) ([const scalar\\_t x](#))
- static INLINE CONST [vect\\_t](#) [set](#) ([const scalar\\_t x0](#), [const scalar\\_t x1](#))
- template<class [T](#) >  
static INLINE PURE [vect\\_t](#) [gather](#) ([const scalar\\_t \\*const p](#), [const T \\*const idx](#))
- template<int [idx](#)>  
static INLINE CONST [scalar\\_t](#) [get](#) ([vect\\_t v](#))
- static INLINE PURE [vect\\_t](#) [load](#) ([const scalar\\_t \\*const p](#))
- static INLINE PURE [vect\\_t](#) [loadu](#) ([const scalar\\_t \\*const p](#))
- static INLINE void [store](#) ([scalar\\_t \\*p](#), [vect\\_t v](#))
- static INLINE void [storeu](#) ([scalar\\_t \\*p](#), [vect\\_t v](#))
- static INLINE void [stream](#) ([scalar\\_t \\*p](#), [const vect\\_t v](#))
- template<int [s](#)>  
static INLINE CONST [vect\\_t](#) [sll](#) ([const vect\\_t a](#))
- template<int [s](#)>  
static INLINE CONST [vect\\_t](#) [srl](#) ([const vect\\_t a](#))
- template<int [s](#)>  
static INLINE CONST [vect\\_t](#) [sra](#) ([const vect\\_t a](#))
- template<uint8\_t [s](#)>  
static INLINE CONST [vect\\_t](#) [shuffle](#) ([const vect\\_t a](#))
- static INLINE CONST [vect\\_t](#) [unpacklo\\_intrinsic](#) ([const vect\\_t a](#), [const vect\\_t b](#))
- static INLINE CONST [vect\\_t](#) [unpackhi\\_intrinsic](#) ([const vect\\_t a](#), [const vect\\_t b](#))
- static INLINE CONST [vect\\_t](#) [unpacklo](#) ([const vect\\_t a](#), [const vect\\_t b](#))
- static INLINE CONST [vect\\_t](#) [unpackhi](#) ([const vect\\_t a](#), [const vect\\_t b](#))
- static INLINE void [unpacklohi](#) ([vect\\_t &lo](#), [vect\\_t &hi](#), [const vect\\_t a](#), [const vect\\_t b](#))
- static INLINE CONST [vect\\_t](#) [pack\\_even](#) ([const vect\\_t a](#), [const vect\\_t b](#))
- static INLINE CONST [vect\\_t](#) [pack\\_odd](#) ([const vect\\_t a](#), [const vect\\_t b](#))
- static INLINE void [pack](#) ([vect\\_t &even](#), [vect\\_t &odd](#), [const vect\\_t a](#), [const vect\\_t b](#))
- static INLINE void [transpose](#) ([vect\\_t &r0](#), [vect\\_t &r1](#))

- `template<uint8_t s>`  
`static INLINE CONST vect_t blend (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t add (const vect_t a, const vect_t b)`
- `static INLINE vect_t addin (vect_t &a, const vect_t b)`
- `static INLINE CONST vect_t sub (const vect_t a, const vect_t b)`
- `static INLINE vect_t subin (vect_t &a, const vect_t b)`
- `static INLINE CONST vect_t mullo (const vect_t x0, const vect_t x1)`
- `static INLINE CONST vect_t mul (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t mulhi (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t mulx (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmadd (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmaddin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmaddx (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmaddxin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fnmadd (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fnmaddin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fnmaddx (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fnmaddxin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmsub (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmsubin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmsubx (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmsubxin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t eq (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t greater (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t lesser (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t greater_eq (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t lesser_eq (const vect_t a, const vect_t b)`
- `static INLINE CONST scalar_t hadd_to_scal (const vect_t a)`
- `static INLINE CONST vect_t round (const vect_t a)`
- `static INLINE CONST vect_t mask_high ()`
- `static INLINE CONST vect_t mulhi_fast (vect_t x, vect_t y)`
- `static INLINE vect_t mod (vect_t &C, const __m128d &P, const __m128d &INVP, const __m128d &NEGP, const vect_t &POW50REM, const __m128d &MIN, const __m128d &MAX, __m128d &Q, __m128d &T)`
- `static INLINE CONST vect_t zero ()`
- `template<uint8_t s>`  
`static INLINE CONST vect_t sll128 (const vect_t a)`
- `template<uint8_t s>`  
`static INLINE CONST vect_t srl128 (const vect_t a)`
- `static INLINE CONST vect_t vand (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t vor (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t vxor (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t vandnot (const vect_t a, const vect_t b)`

### Static Public Attributes

- `static const constexpr size_t vect_size = 2`
- `static const constexpr size_t alignment = 16`

### Static Protected Member Functions

- `static INLINE CONST vect_t signbits (const vect_t x)`

## 12.280.1 Member Typedef Documentation

### 12.280.1.1 vect\_t

```
using vect_t = __m128i
```

**12.280.1.2 scalar\_t**

```
using scalar_t = int64_t
```

**12.280.1.3 aligned\_allocator**

```
using aligned_allocator = AlignedAllocator<scalar_t, Alignment(alignment)>
```

**12.280.1.4 aligned\_vector**

```
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```

**12.280.1.5 is\_same\_element**

```
template<class Field >
using is_same_element = std::is_same<typename Field::Element, scalar_t>
```

**12.280.2 Member Function Documentation****12.280.2.1 type\_string()**

```
static const std::string type_string () [inline], [static]
```

**12.280.2.2 valid()**

```
template<class T >
static constexpr bool valid (
 T * p) [inline], [static], [constexpr]
```

**12.280.2.3 compliant()**

```
template<class T >
static constexpr bool compliant (
 T n) [inline], [static], [constexpr]
```

**12.280.2.4 set1()**

```
static INLINE CONST vect_t set1 (
 const scalar_t x) [inline], [static]
```

**12.280.2.5 set()**

```
static INLINE CONST vect_t set (
 const scalar_t x0,
 const scalar_t x1) [inline], [static]
```

**12.280.2.6 gather()**

```
template<class T >
static INLINE PURE vect_t gather (
 const scalar_t *const p,
 const T *const idx) [inline], [static]
```

**12.280.2.7 get()**

```
template<int idx>
static INLINE CONST scalar_t get (
 vect_t v) [inline], [static]
```

**12.280.2.8 load()**

```
static INLINE PURE vect_t load (
 const scalar_t *const p) [inline], [static]
```

**12.280.2.9 loadu()**

```
static INLINE PURE vect_t loadu (
 const scalar_t *const p) [inline], [static]
```

**12.280.2.10 store()**

```
static INLINE void store (
 scalar_t * p,
 vect_t v) [inline], [static]
```

**12.280.2.11 storeu()**

```
static INLINE void storeu (
 scalar_t * p,
 vect_t v) [inline], [static]
```

**12.280.2.12 stream()**

```
static INLINE void stream (
 scalar_t * p,
 const vect_t v) [inline], [static]
```

**12.280.2.13 sll()**

```
template<int s>
static INLINE CONST vect_t sll (
 const vect_t a) [inline], [static]
```

**12.280.2.14 srl()**

```
template<int s>
static INLINE CONST vect_t srl (
 const vect_t a) [inline], [static]
```

**12.280.2.15 sra()**

```
template<int s>
static INLINE CONST vect_t sra (
 const vect_t a) [inline], [static]
```

**12.280.2.16 shuffle()**

```
template<uint8_t s>
static INLINE CONST vect_t shuffle (
 const vect_t a) [inline], [static]
```

**12.280.2.17 unpacklo\_intrinsic()**

```
static INLINE CONST vect_t unpacklo_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.280.2.18 unpackhi\_intrinsic()**

```
static INLINE CONST vect_t unpackhi_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.280.2.19 unpacklo()**

```
static INLINE CONST vect_t unpacklo (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.280.2.20 unpackhi()**

```
static INLINE CONST vect_t unpackhi (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.280.2.21 unpacklohi()**

```
static INLINE void unpacklohi (
 vect_t & lo,
 vect_t & hi,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.280.2.22 pack\_even()**

```
static INLINE CONST vect_t pack_even (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.280.2.23 pack\_odd()**

```
static INLINE CONST vect_t pack_odd (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.280.2.24 pack()**

```
static INLINE void pack (
 vect_t & even,
 vect_t & odd,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.280.2.25 transpose()**

```
static INLINE void transpose (
 vect_t & r0,
 vect_t & r1) [inline], [static]
```

**12.280.2.26 blend()**

```
template<uint8_t s>
static INLINE CONST vect_t blend (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.280.2.27 add()**

```
static INLINE CONST vect_t add (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.280.2.28 addin()**

```
static INLINE vect_t addin (
 vect_t & a,
 const vect_t b) [inline], [static]
```

**12.280.2.29 sub()**

```
static INLINE CONST vect_t sub (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.280.2.30 subin()**

```
static INLINE vect_t subin (
 vect_t & a,
 const vect_t b) [inline], [static]
```

**12.280.2.31 mullo()**

```
static INLINE CONST vect_t mullo (
 const vect_t x0,
 const vect_t x1) [inline], [static]
```

**12.280.2.32 mul()**

```
static INLINE CONST vect_t mul (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.280.2.33 mulhi()**

```
static INLINE CONST vect_t mulhi (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.280.2.34 mulx()**

```
static INLINE CONST vect_t mulx (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.280.2.35 fmadd()**

```
static INLINE CONST vect_t fmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.280.2.36 fmaddin()**

```
static INLINE vect_t fmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.280.2.37 fmaddx()**

```
static INLINE CONST vect_t fmaddx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.280.2.38 fmaddxin()**

```
static INLINE vect_t fmaddxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.280.2.39 fnmadd()**

```
static INLINE CONST vect_t fnmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.280.2.40 fnmaddin()**

```
static INLINE vect_t fnmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.280.2.41 fnmaddx()**

```
static INLINE CONST vect_t fnmaddx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.280.2.42 fnmaddxin()**

```
static INLINE vect_t fnmaddxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.280.2.43 fmsub()**

```
static INLINE CONST vect_t fmsub (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.280.2.44 fmsubin()**

```
static INLINE vect_t fmsubin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.280.2.45 fmsubx()**

```
static INLINE CONST vect_t fmsubx (
 const vect_t c,
```

```
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.280.2.46 fmsubxin()

```
static INLINE vect_t fmsubxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.280.2.47 eq()

```
static INLINE CONST vect_t eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.280.2.48 greater()

```
static INLINE CONST vect_t greater (
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.280.2.49 lesser()

```
static INLINE CONST vect_t lesser (
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.280.2.50 greater\_eq()

```
static INLINE CONST vect_t greater_eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.280.2.51 lesser\_eq()

```
static INLINE CONST vect_t lesser_eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.280.2.52 hadd\_to\_scal()

```
static INLINE CONST scalar_t hadd_to_scal (
 const vect_t a) [inline], [static]
```

#### 12.280.2.53 round()

```
static INLINE CONST vect_t round (
 const vect_t a) [inline], [static]
```

#### 12.280.2.54 mask\_high()

```
static INLINE CONST vect_t mask_high () [inline], [static]
```

#### 12.280.2.55 mulhi\_fast()

```
INLINE CONST vect_t mulhi_fast (
 vect_t x,
 vect_t y) [static]
```



**12.280.2.56 mod()**

```

INLINE vect_t mod (
 vect_t & C,
 const __m128d & P,
 const __m128d & INV_P,
 const __m128d & NEG_P,
 const vect_t & POW50REM,
 const __m128d & MIN,
 const __m128d & MAX,
 __m128d & Q,
 __m128d & T) [static]

```

**12.280.2.57 signbits()**

```

static INLINE CONST vect_t signbits (
 const vect_t x) [inline], [static], [protected]

```

**12.280.2.58 zero()**

```

static INLINE CONST vect_t zero () [inline], [static], [inherited]

```

**12.280.2.59 sll128()**

```

template<uint8_t s>
static INLINE CONST vect_t sll128 (
 const vect_t a) [inline], [static], [inherited]

```

**12.280.2.60 srl128()**

```

template<uint8_t s>
static INLINE CONST vect_t srl128 (
 const vect_t a) [inline], [static], [inherited]

```

**12.280.2.61 vand()**

```

static INLINE CONST vect_t vand (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]

```

**12.280.2.62 vor()**

```

static INLINE CONST vect_t vor (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]

```

**12.280.2.63 vxor()**

```

static INLINE CONST vect_t vxor (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]

```

**12.280.2.64 vandnot()**

```

static INLINE CONST vect_t vandnot (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]

```

## 12.280.3 Field Documentation

### 12.280.3.1 vect\_size

```
const constexpr size_t vect_size = 2 [static], [constexpr]
```

### 12.280.3.2 alignment

```
const constexpr size_t alignment = 16 [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [simd128\\_int64.inl](#)

## 12.281 Simd128i\_base Struct Reference

Inheritance diagram for Simd128i\_base:



### Public Types

- `using vect_t = __m128i`

### Static Public Member Functions

- `static INLINE CONST vect_t zero ()`
- `template<uint8_t s>`  
`static INLINE CONST vect_t sll128 (const vect_t a)`
- `template<uint8_t s>`  
`static INLINE CONST vect_t srl128 (const vect_t a)`
- `static INLINE CONST vect_t vand (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t vor (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t vxor (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t vandnot (const vect_t a, const vect_t b)`

## 12.281.1 Member Typedef Documentation

### 12.281.1.1 vect\_t

```
using vect_t = __m128i
```

## 12.281.2 Member Function Documentation

### 12.281.2.1 zero()

```
static INLINE CONST vect_t zero () [inline], [static]
```

### 12.281.2.2 sll128()

```
template<uint8_t s>
static INLINE CONST vect_t sll128 (
 const vect_t a) [inline], [static]
```

**12.281.2.3 srl128()**

```
template<uint8_t s>
static INLINE CONST vect_t srl128 (
 const vect_t a) [inline], [static]
```

**12.281.2.4 vand()**

```
static INLINE CONST vect_t vand (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.281.2.5 vor()**

```
static INLINE CONST vect_t vor (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.281.2.6 vxor()**

```
static INLINE CONST vect_t vxor (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.281.2.7 vandnot()**

```
static INLINE CONST vect_t vandnot (
 const vect_t a,
 const vect_t b) [inline], [static]
```

The documentation for this struct was generated from the following file:

- [simd128.inl](#)

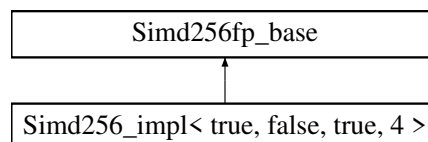
**12.282 Simd256\_impl< ArithType, Int, Signed, Size > Struct Template Reference**

The documentation for this struct was generated from the following file:

- [simd256.inl](#)

**12.283 Simd256\_impl< true, false, true, 4 > Struct Reference**

Inheritance diagram for Simd256\_impl< true, false, true, 4 >:

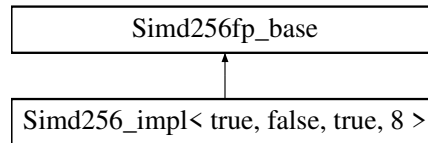


The documentation for this struct was generated from the following file:

- [simd256\\_float.inl](#)

## 12.284 Simd256\_impl< true, false, true, 8 > Struct Reference

Inheritance diagram for Simd256\_impl< true, false, true, 8 >:



### Data Structures

- union [Converter](#)

### Public Types

- using [vect\\_t](#) = [\\_\\_m256d](#)
- using [scalar\\_t](#) = [double](#)
- using [aligned\\_allocator](#) = [AlignedAllocator](#)< [scalar\\_t](#), [Alignment](#)([alignment](#))>
- using [aligned\\_vector](#) = [std::vector](#)< [scalar\\_t](#), [aligned\\_allocator](#) >
- template<class [Field](#) >  
using [is\\_same\\_element](#) = [std::is\\_same](#)< [typename Field::Element](#), [scalar\\_t](#) >

### Static Public Member Functions

- static const [std::string](#) [type\\_string](#) ()
- template<class [T](#) >  
static constexpr bool [valid](#) ([T](#) \*[p](#))
- template<class [T](#) >  
static constexpr bool [compliant](#) ([T](#) [n](#))
- static INLINE CONST [vect\\_t](#) [zero](#) ()
- static INLINE CONST [vect\\_t](#) [set1](#) (const [scalar\\_t](#) [x](#))
- static INLINE CONST [vect\\_t](#) [set](#) (const [scalar\\_t](#) [x1](#), const [scalar\\_t](#) [x2](#), const [scalar\\_t](#) [x3](#), const [scalar\\_t](#) [x4](#))
- template<class [T](#) >  
static INLINE PURE [vect\\_t](#) [gather](#) (const [scalar\\_t](#) \*const [p](#), const [T](#) \*const [idx](#))
- static INLINE PURE [vect\\_t](#) [load](#) (const [scalar\\_t](#) \*const [p](#))
- static INLINE PURE [vect\\_t](#) [loadu](#) (const [scalar\\_t](#) \*const [p](#))
- static INLINE void [store](#) (const [scalar\\_t](#) \*[p](#), const [vect\\_t](#) [v](#))
- static INLINE void [storeu](#) (const [scalar\\_t](#) \*[p](#), const [vect\\_t](#) [v](#))
- static INLINE void [stream](#) (const [scalar\\_t](#) \*[p](#), const [vect\\_t](#) [v](#))
- static INLINE CONST [vect\\_t](#) [unpacklo\\_intrinsic](#) (const [vect\\_t](#) [a](#), const [vect\\_t](#) [b](#))
- static INLINE CONST [vect\\_t](#) [unpackhi\\_intrinsic](#) (const [vect\\_t](#) [a](#), const [vect\\_t](#) [b](#))
- static INLINE CONST [vect\\_t](#) [unpacklo](#) (const [vect\\_t](#) [a](#), const [vect\\_t](#) [b](#))
- static INLINE CONST [vect\\_t](#) [unpackhi](#) (const [vect\\_t](#) [a](#), const [vect\\_t](#) [b](#))
- static INLINE void [unpacklohi](#) ([vect\\_t](#) &[lo](#), [vect\\_t](#) &[hi](#), const [vect\\_t](#) [a](#), const [vect\\_t](#) [b](#))
- static INLINE CONST [vect\\_t](#) [pack\\_even](#) (const [vect\\_t](#) [a](#), const [vect\\_t](#) [b](#))
- static INLINE CONST [vect\\_t](#) [pack\\_odd](#) (const [vect\\_t](#) [a](#), const [vect\\_t](#) [b](#))
- static INLINE void [pack](#) ([vect\\_t](#) &[even](#), [vect\\_t](#) &[odd](#), const [vect\\_t](#) [a](#), const [vect\\_t](#) [b](#))
- static INLINE void [transpose](#) ([vect\\_t](#) &[r0](#), [vect\\_t](#) &[r1](#), [vect\\_t](#) &[r2](#), [vect\\_t](#) &[r3](#))
- template<uint8\_t [s](#)>  
static INLINE CONST [vect\\_t](#) [blend](#) (const [vect\\_t](#) [a](#), const [vect\\_t](#) [b](#))
- static INLINE CONST [vect\\_t](#) [blendv](#) (const [vect\\_t](#) [a](#), const [vect\\_t](#) [b](#), const [vect\\_t](#) [mask](#))
- static INLINE CONST [vect\\_t](#) [add](#) (const [vect\\_t](#) [a](#), const [vect\\_t](#) [b](#))
- static INLINE [vect\\_t](#) [addin](#) ([vect\\_t](#) &[a](#), const [vect\\_t](#) [b](#))
- static INLINE CONST [vect\\_t](#) [sub](#) (const [vect\\_t](#) [a](#), const [vect\\_t](#) [b](#))
- static INLINE CONST [vect\\_t](#) [subin](#) ([vect\\_t](#) &[a](#), const [vect\\_t](#) [b](#))

- static `INLINE CONST vect_t mul (const vect_t a, const vect_t b)`
- static `INLINE CONST vect_t mulin (vect_t &a, const vect_t b)`
- static `INLINE CONST vect_t div (const vect_t a, const vect_t b)`
- static `INLINE CONST vect_t fmadd (const vect_t c, const vect_t a, const vect_t b)`
- static `INLINE CONST vect_t fmaddin (vect_t &c, const vect_t a, const vect_t b)`
- static `INLINE CONST vect_t fnmadd (const vect_t c, const vect_t a, const vect_t b)`
- static `INLINE CONST vect_t fnmaddin (vect_t &c, const vect_t a, const vect_t b)`
- static `INLINE CONST vect_t fmsub (const vect_t c, const vect_t a, const vect_t b)`
- static `INLINE CONST vect_t fmsubin (vect_t &c, const vect_t a, const vect_t b)`
- static `INLINE CONST vect_t eq (const vect_t a, const vect_t b)`
- static `INLINE CONST vect_t lesser (const vect_t a, const vect_t b)`
- static `INLINE CONST vect_t lesser_eq (const vect_t a, const vect_t b)`
- static `INLINE CONST vect_t greater (const vect_t a, const vect_t b)`
- static `INLINE CONST vect_t greater_eq (const vect_t a, const vect_t b)`
- static `INLINE CONST vect_t vand (const vect_t a, const vect_t b)`
- static `INLINE CONST vect_t vor (const vect_t a, const vect_t b)`
- static `INLINE CONST vect_t vxor (const vect_t a, const vect_t b)`
- static `INLINE CONST vect_t vandnot (const vect_t a, const vect_t b)`
- static `INLINE CONST vect_t floor (const vect_t a)`
- static `INLINE CONST vect_t ceil (const vect_t a)`
- static `INLINE CONST vect_t round (const vect_t a)`
- static `INLINE CONST vect_t hadd (const vect_t a, const vect_t b)`
- static `INLINE CONST scalar_t hadd_to_scal (const vect_t a)`
- static `INLINE vect_t mod (vect_t &C, const vect_t &P, const vect_t &INVP, const vect_t &NEGP, const vect_t &MIN, const vect_t &MAX, vect_t &Q, vect_t &T)`

#### Static Public Attributes

- static `const constexpr size_t vect_size = 4`
- static `const constexpr size_t alignment = 32`

### 12.284.1 Member Typedef Documentation

#### 12.284.1.1 vect\_t

```
using vect_t = __m256d
```

#### 12.284.1.2 scalar\_t

```
using scalar_t = double
```

#### 12.284.1.3 aligned\_allocator

```
using aligned_allocator = AlignedAllocator<scalar_t, Alignment(alignment)>
```

#### 12.284.1.4 aligned\_vector

```
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```

#### 12.284.1.5 is\_same\_element

```
template<class Field >
using is_same_element = std::is_same<typename Field::Element, scalar_t>
```

### 12.284.2 Member Function Documentation

#### 12.284.2.1 type\_string()

```
static const std::string type_string () [inline], [static]
```

#### 12.284.2.2 valid()

```
template<class T >
static constexpr bool valid (
 T * p) [inline], [static], [constexpr]
```

#### 12.284.2.3 compliant()

```
template<class T >
static constexpr bool compliant (
 T n) [inline], [static], [constexpr]
```

#### 12.284.2.4 zero()

```
static INLINE CONST vect_t zero () [inline], [static]
```

#### 12.284.2.5 set1()

```
static INLINE CONST vect_t set1 (
 const scalar_t x) [inline], [static]
```

#### 12.284.2.6 set()

```
static INLINE CONST vect_t set (
 const scalar_t x1,
 const scalar_t x2,
 const scalar_t x3,
 const scalar_t x4) [inline], [static]
```

#### 12.284.2.7 gather()

```
template<class T >
static INLINE PURE vect_t gather (
 const scalar_t *const p,
 const T *const idx) [inline], [static]
```

#### 12.284.2.8 load()

```
static INLINE PURE vect_t load (
 const scalar_t *const p) [inline], [static]
```

#### 12.284.2.9 loadu()

```
static INLINE PURE vect_t loadu (
 const scalar_t *const p) [inline], [static]
```

#### 12.284.2.10 store()

```
static INLINE void store (
 const scalar_t * p,
 const vect_t v) [inline], [static]
```

#### 12.284.2.11 storeu()

```
static INLINE void storeu (
 const scalar_t * p,
 const vect_t v) [inline], [static]
```

**12.284.2.12 stream()**

```
static INLINE void stream (
 const scalar_t * p,
 const vect_t v) [inline], [static]
```

**12.284.2.13 unpacklo\_intrinsic()**

```
static INLINE CONST vect_t unpacklo_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.284.2.14 unpackhi\_intrinsic()**

```
static INLINE CONST vect_t unpackhi_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.284.2.15 unpacklo()**

```
static INLINE CONST vect_t unpacklo (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.284.2.16 unpackhi()**

```
static INLINE CONST vect_t unpackhi (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.284.2.17 unpacklohi()**

```
static INLINE void unpacklohi (
 vect_t & lo,
 vect_t & hi,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.284.2.18 pack\_even()**

```
static INLINE CONST vect_t pack_even (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.284.2.19 pack\_odd()**

```
static INLINE CONST vect_t pack_odd (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.284.2.20 pack()**

```
static INLINE void pack (
 vect_t & even,
 vect_t & odd,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.284.2.21 transpose()**

```
static INLINE void transpose (
 vect_t & r0,
 vect_t & r1,
 vect_t & r2,
 vect_t & r3) [inline], [static]
```

**12.284.2.22 blend()**

```
template<uint8_t s>
static INLINE CONST vect_t blend (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.284.2.23 blendv()**

```
static INLINE CONST vect_t blendv (
 const vect_t a,
 const vect_t b,
 const vect_t mask) [inline], [static]
```

**12.284.2.24 add()**

```
static INLINE CONST vect_t add (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.284.2.25 addin()**

```
static INLINE vect_t addin (
 vect_t & a,
 const vect_t b) [inline], [static]
```

**12.284.2.26 sub()**

```
static INLINE CONST vect_t sub (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.284.2.27 subin()**

```
static INLINE CONST vect_t subin (
 vect_t & a,
 const vect_t b) [inline], [static]
```

**12.284.2.28 mul()**

```
static INLINE CONST vect_t mul (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.284.2.29 mulin()**

```
static INLINE CONST vect_t mulin (
 vect_t & a,
 const vect_t b) [inline], [static]
```



**12.284.2.30 div()**

```
static INLINE CONST vect_t div (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.284.2.31 fmadd()**

```
static INLINE CONST vect_t fmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.284.2.32 fmaddin()**

```
static INLINE CONST vect_t fmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.284.2.33 fnmadd()**

```
static INLINE CONST vect_t fnmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.284.2.34 fnmaddin()**

```
static INLINE CONST vect_t fnmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.284.2.35 fmsub()**

```
static INLINE CONST vect_t fmsub (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.284.2.36 fmsubin()**

```
static INLINE CONST vect_t fmsubin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.284.2.37 eq()**

```
static INLINE CONST vect_t eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.284.2.38 lesser()**

```
static INLINE CONST vect_t lesser (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.284.2.39 lesser\_eq()**

```
static INLINE CONST vect_t lesser_eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.284.2.40 greater()**

```
static INLINE CONST vect_t greater (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.284.2.41 greater\_eq()**

```
static INLINE CONST vect_t greater_eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.284.2.42 vand()**

```
static INLINE CONST vect_t vand (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.284.2.43 vor()**

```
static INLINE CONST vect_t vor (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.284.2.44 vxor()**

```
static INLINE CONST vect_t vxor (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.284.2.45 vandnot()**

```
static INLINE CONST vect_t vandnot (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.284.2.46 floor()**

```
static INLINE CONST vect_t floor (
 const vect_t a) [inline], [static]
```

**12.284.2.47 ceil()**

```
static INLINE CONST vect_t ceil (
 const vect_t a) [inline], [static]
```

**12.284.2.48 round()**

```
static INLINE CONST vect_t round (
 const vect_t a) [inline], [static]
```

**12.284.2.49 hadd()**

```
static INLINE CONST vect_t hadd (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.284.2.50 hadd\_to\_scal()**

```
static INLINE CONST scalar_t hadd_to_scal (
 const vect_t a) [inline], [static]
```

**12.284.2.51 mod()**

```
static INLINE vect_t mod (
 vect_t & C,
 const vect_t & P,
 const vect_t & INV_P,
 const vect_t & NEG_P,
 const vect_t & MIN,
 const vect_t & MAX,
 vect_t & Q,
 vect_t & T) [inline], [static]
```

**12.284.3 Field Documentation****12.284.3.1 vect\_size**

```
const constexpr size_t vect_size = 4 [static], [constexpr]
```

**12.284.3.2 alignment**

```
const constexpr size_t alignment = 32 [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [simd256\\_double.inl](#)

**12.285 Simd256\_impl< true, true, false, 2 > Struct Reference**

Inheritance diagram for Simd256\_impl< true, true, false, 2 >:

**Data Structures**

- union [Converter](#)

**Public Types**

- using [scalar\\_t](#) = [uint16\\_t](#)
- using [aligned\\_allocator](#) = [AlignedAllocator](#)< [scalar\\_t](#), [Alignment](#)([alignment](#))>
- using [aligned\\_vector](#) = [std::vector](#)< [scalar\\_t](#), [aligned\\_allocator](#) >

- `template<class Field >`  
`using is_same_element = std::is_same< typename Field::Element, scalar_t >`
- `using simdHalf = Simd128< scalar_t >`
- `using vect_t = __m256i`
- `using half_t = __m128i`

### Static Public Member Functions

- `static const std::string type_string ()`
- `static INLINE CONST vect_t set1 (const scalar_t x)`
- `static INLINE CONST vect_t set (const scalar_t x0, const scalar_t x1, const scalar_t x2, const scalar_t x3, const scalar_t x4, const scalar_t x5, const scalar_t x6, const scalar_t x7, const scalar_t x8, const scalar_t x9, const scalar_t x10, const scalar_t x11, const scalar_t x12, const scalar_t x13, const scalar_t x14, const scalar_t x15)`
- `template<class T >`  
`static INLINE PURE vect_t gather (const scalar_t *const p, const T *const idx)`
- `static INLINE PURE vect_t load (const scalar_t *const p)`
- `static INLINE PURE vect_t loadu (const scalar_t *const p)`
- `static INLINE void store (scalar_t *p, vect_t v)`
- `static INLINE void storeu (scalar_t *p, vect_t v)`
- `static INLINE void stream (scalar_t *p, const vect_t v)`
- `template<int s>`  
`static INLINE CONST vect_t sra (const vect_t a)`
- `static INLINE CONST vect_t greater (vect_t a, vect_t b)`
- `static INLINE CONST vect_t lesser (vect_t a, vect_t b)`
- `static INLINE CONST vect_t greater_eq (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t lesser_eq (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t mulhi (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t mulx (vect_t a, vect_t b)`
- `static INLINE CONST vect_t fmaddx (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmaddxin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fnmaddx (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fnmaddxin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmsubx (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmsubxin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST scalar_t hadd_to_scal (const vect_t a)`
- `template<class T >`  
`static constexpr bool valid (T *p)`
- `template<class T >`  
`static constexpr bool compliant (T n)`
- `static INLINE CONST vect_t set1 (const scalar_t x)`
- `static INLINE CONST vect_t set (const scalar_t x0, const scalar_t x1, const scalar_t x2, const scalar_t x3, const scalar_t x4, const scalar_t x5, const scalar_t x6, const scalar_t x7, const scalar_t x8, const scalar_t x9, const scalar_t x10, const scalar_t x11, const scalar_t x12, const scalar_t x13, const scalar_t x14, const scalar_t x15)`
- `template<class T >`  
`static INLINE PURE vect_t gather (const scalar_t *const p, const T *const idx)`
- `static INLINE PURE vect_t load (const scalar_t *const p)`
- `static INLINE PURE vect_t loadu (const scalar_t *const p)`
- `static INLINE void store (scalar_t *p, vect_t v)`
- `static INLINE void storeu (scalar_t *p, vect_t v)`
- `static INLINE void stream (scalar_t *p, const vect_t v)`
- `template<int s>`  
`static INLINE CONST vect_t sll (const vect_t a)`
- `template<int s>`  
`static INLINE CONST vect_t srl (const vect_t a)`

- `template<uint64_t s>`  
`static INLINE CONST vect_t shuffle (const vect_t a)`
- `static INLINE CONST vect_t unpacklo_intrinsic (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t unpackhi_intrinsic (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t unpacklo (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t unpackhi (const vect_t a, const vect_t b)`
- `static INLINE void unpacklohi (vect_t &lo, vect_t &hi, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t pack_even (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t pack_odd (const vect_t a, const vect_t b)`
- `static INLINE void pack (vect_t &even, vect_t &odd, const vect_t a, const vect_t b)`
- `static INLINE void transpose (vect_t &r0, vect_t &r1, vect_t &r2, vect_t &r3, vect_t &r4, vect_t &r5, vect_t &r6, vect_t &r7, vect_t &r8, vect_t &r9, vect_t &r10, vect_t &r11, vect_t &r12, vect_t &r13, vect_t &r14, vect_t &r15)`
- `template<uint16_t s, typename std::enable_if<(s & 0x00ff) == (s > > 8)>::type * = nullptr>`  
`static INLINE vect_t blend (const vect_t a, const vect_t b)`
- `template<uint16_t s, typename std::enable_if<(s & 0x00ff) != (s > > 8)>::type * = nullptr>`  
`static INLINE vect_t blend (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t add (const vect_t a, const vect_t b)`
- `static INLINE vect_t addin (vect_t &a, const vect_t b)`
- `static INLINE CONST vect_t sub (const vect_t a, const vect_t b)`
- `static INLINE vect_t subin (vect_t &a, const vect_t b)`
- `static INLINE CONST vect_t mullo (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t mul (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmadd (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmaddin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fnmadd (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fnmaddin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmsub (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmsubin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t eq (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t round (const vect_t a)`
- `static INLINE vect_t mod (vect_t &C, const vect_t &P, const vect_t &INVP, const vect_t &NEGP, const vect_t &MIN, const vect_t &MAX, vect_t &Q, vect_t &T)`
- `static INLINE CONST vect_t zero ()`

### Static Public Attributes

- `static const constexpr size_t vect_size = 16`
- `static const constexpr size_t alignment = 32`

## 12.285.1 Member Typedef Documentation

### 12.285.1.1 scalar\_t

```
using scalar_t = uint16_t
```

### 12.285.1.2 aligned\_allocator

```
using aligned_allocator = AlignedAllocator<scalar_t, Alignment(alignment)>
```

### 12.285.1.3 aligned\_vector

```
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```

### 12.285.1.4 is\_same\_element

```
template<class Field >
using is_same_element = std::is_same<typename Field::Element, scalar_t>
```

**12.285.1.5 simdHalf**

```
using simdHalf = Simd128<scalar_t>
```

**12.285.1.6 vect\_t**

```
using vect_t = __m256i [inherited]
```

**12.285.1.7 half\_t**

```
using half_t = __m128i [inherited]
```

**12.285.2 Member Function Documentation****12.285.2.1 type\_string()**

```
static const std::string type_string () [inline], [static]
```

**12.285.2.2 set1() [1/2]**

```
static INLINE CONST vect_t set1 (
 const scalar_t x) [inline], [static]
```

**12.285.2.3 set() [1/2]**

```
static INLINE CONST vect_t set (
 const scalar_t x0,
 const scalar_t x1,
 const scalar_t x2,
 const scalar_t x3,
 const scalar_t x4,
 const scalar_t x5,
 const scalar_t x6,
 const scalar_t x7,
 const scalar_t x8,
 const scalar_t x9,
 const scalar_t x10,
 const scalar_t x11,
 const scalar_t x12,
 const scalar_t x13,
 const scalar_t x14,
 const scalar_t x15) [inline], [static]
```

**12.285.2.4 gather() [1/2]**

```
template<class T >
static INLINE PURE vect_t gather (
 const scalar_t *const p,
 const T *const idx) [inline], [static]
```

**12.285.2.5 load() [1/2]**

```
static INLINE PURE vect_t load (
 const scalar_t *const p) [inline], [static]
```

**12.285.2.6 loadu() [1/2]**

```
static INLINE PURE vect_t loadu (
 const scalar_t *const p) [inline], [static]
```

**12.285.2.7 store()** [1/2]

```
static INLINE void store (
 scalar_t * p,
 vect_t v) [inline], [static]
```

**12.285.2.8 storeu()** [1/2]

```
static INLINE void storeu (
 scalar_t * p,
 vect_t v) [inline], [static]
```

**12.285.2.9 stream()** [1/2]

```
static INLINE void stream (
 scalar_t * p,
 const vect_t v) [inline], [static]
```

**12.285.2.10 sra()**

```
template<int s>
static INLINE CONST vect_t sra (
 const vect_t a) [inline], [static]
```

**12.285.2.11 greater()**

```
static INLINE CONST vect_t greater (
 vect_t a,
 vect_t b) [inline], [static]
```

**12.285.2.12 lesser()**

```
static INLINE CONST vect_t lesser (
 vect_t a,
 vect_t b) [inline], [static]
```

**12.285.2.13 greater\_eq()**

```
static INLINE CONST vect_t greater_eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.285.2.14 lesser\_eq()**

```
static INLINE CONST vect_t lesser_eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.285.2.15 mulhi()**

```
static INLINE CONST vect_t mulhi (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.285.2.16 mulx()**

```
static INLINE CONST vect_t mulx (
 vect_t a,
 vect_t b) [inline], [static]
```

**12.285.2.17 fmaddx()**

```
static INLINE CONST vect_t fmaddx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.285.2.18 fmaddxin()**

```
static INLINE vect_t fmaddxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.285.2.19 fnmaddx()**

```
static INLINE CONST vect_t fnmaddx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.285.2.20 fnmaddxin()**

```
static INLINE vect_t fnmaddxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.285.2.21 fmsubx()**

```
static INLINE CONST vect_t fmsubx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.285.2.22 fmsubxin()**

```
static INLINE vect_t fmsubxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.285.2.23 hadd\_to\_scal()**

```
static INLINE CONST scalar_t hadd_to_scal (
 const vect_t a) [inline], [static]
```

**12.285.2.24 valid()**

```
template<class T >
static constexpr bool valid (
 T * p) [inline], [static], [constexpr], [inherited]
```

**12.285.2.25 compliant()**

```
template<class T >
static constexpr bool compliant (
 T n) [inline], [static], [constexpr], [inherited]
```



**12.285.2.26 set1()** [2/2]

```
static INLINE CONST vect_t set1 (
 const scalar_t x) [inline], [static], [inherited]
```

**12.285.2.27 set()** [2/2]

```
static INLINE CONST vect_t set (
 const scalar_t x0,
 const scalar_t x1,
 const scalar_t x2,
 const scalar_t x3,
 const scalar_t x4,
 const scalar_t x5,
 const scalar_t x6,
 const scalar_t x7,
 const scalar_t x8,
 const scalar_t x9,
 const scalar_t x10,
 const scalar_t x11,
 const scalar_t x12,
 const scalar_t x13,
 const scalar_t x14,
 const scalar_t x15) [inline], [static], [inherited]
```

**12.285.2.28 gather()** [2/2]

```
template<class T >
static INLINE PURE vect_t gather (
 const scalar_t *const p,
 const T *const idx) [inline], [static], [inherited]
```

**12.285.2.29 load()** [2/2]

```
static INLINE PURE vect_t load (
 const scalar_t *const p) [inline], [static], [inherited]
```

**12.285.2.30 loadu()** [2/2]

```
static INLINE PURE vect_t loadu (
 const scalar_t *const p) [inline], [static], [inherited]
```

**12.285.2.31 store()** [2/2]

```
static INLINE void store (
 scalar_t * p,
 vect_t v) [inline], [static], [inherited]
```

**12.285.2.32 storeu()** [2/2]

```
static INLINE void storeu (
 scalar_t * p,
 vect_t v) [inline], [static], [inherited]
```

**12.285.2.33 stream()** [2/2]

```
static INLINE void stream (
 scalar_t * p,
 const vect_t v) [inline], [static], [inherited]
```

**12.285.2.34 sll()**

```
template<int s>
static INLINE CONST vect_t sll (
 const vect_t a) [inline], [static], [inherited]
```

**12.285.2.35 srl()**

```
template<int s>
static INLINE CONST vect_t srl (
 const vect_t a) [inline], [static], [inherited]
```

**12.285.2.36 shuffle()**

```
template<uint64_t s>
static INLINE CONST vect_t shuffle (
 const vect_t a) [inline], [static], [inherited]
```

**12.285.2.37 unpacklo\_intrinsic()**

```
static INLINE CONST vect_t unpacklo_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.285.2.38 unpackhi\_intrinsic()**

```
static INLINE CONST vect_t unpackhi_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.285.2.39 unpacklo()**

```
static INLINE CONST vect_t unpacklo (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.285.2.40 unpackhi()**

```
static INLINE CONST vect_t unpackhi (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.285.2.41 unpacklohi()**

```
static INLINE void unpacklohi (
 vect_t & lo,
 vect_t & hi,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.285.2.42 pack\_even()**

```
static INLINE CONST vect_t pack_even (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.285.2.43 pack\_odd()**

```
static INLINE CONST vect_t pack_odd (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.285.2.44 pack()**

```
static INLINE void pack (
 vect_t & even,
 vect_t & odd,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.285.2.45 transpose()**

```
static INLINE void transpose (
 vect_t & r0,
 vect_t & r1,
 vect_t & r2,
 vect_t & r3,
 vect_t & r4,
 vect_t & r5,
 vect_t & r6,
 vect_t & r7,
 vect_t & r8,
 vect_t & r9,
 vect_t & r10,
 vect_t & r11,
 vect_t & r12,
 vect_t & r13,
 vect_t & r14,
 vect_t & r15) [inline], [static], [inherited]
```

**12.285.2.46 blend() [1/2]**

```
template<uint16_t s, typename std::enable_if<(s &0x00ff)==(s > > 8)>::type * = nullptr>
static INLINE vect_t blend (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.285.2.47 blend() [2/2]**

```
template<uint16_t s, typename std::enable_if<(s &0x00ff) !=(s > > 8)>::type * = nullptr>
static INLINE vect_t blend (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.285.2.48 add()**

```
static INLINE CONST vect_t add (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.285.2.49 addin()**

```
static INLINE vect_t addin (
 vect_t & a,
 const vect_t b) [inline], [static], [inherited]
```

**12.285.2.50 sub()**

```
static INLINE CONST vect_t sub (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.285.2.51 subin()**

```
static INLINE vect_t subin (
 vect_t & a,
 const vect_t b) [inline], [static], [inherited]
```

**12.285.2.52 mullo()**

```
static INLINE CONST vect_t mullo (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.285.2.53 mul()**

```
static INLINE CONST vect_t mul (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.285.2.54 fmadd()**

```
static INLINE CONST vect_t fmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.285.2.55 fmaddin()**

```
static INLINE vect_t fmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.285.2.56 fnmadd()**

```
static INLINE CONST vect_t fnmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.285.2.57 fnmaddin()**

```
static INLINE vect_t fnmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.285.2.58 fmsub()**

```
static INLINE CONST vect_t fmsub (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.285.2.59 fmsubin()**

```
static INLINE vect_t fmsubin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.285.2.60** **eq()**

```
static INLINE CONST vect_t eq (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

### 12.285.2.61 round()

```
static INLINE CONST vect_t round (
 const vect_t a) [inline], [static], [inherited]
```

### 12.285.2.62 mod()

```
static INLINE vect_t mod (
 vect_t & C,
 const vect_t & P,
 const vect_t & INVP,
 const vect_t & NEGP,
 const vect_t & MIN,
 const vect_t & MAX,
 vect_t & Q,
 vect_t & T) [inline], [static], [inherited]
```

### 12.285.2.63 zero()

```
static INLINE CONST vect_t zero () [inline], [static], [inherited]
```

### 12.285.3 Field Documentation

### 12.285.3.1 vect\_size

```
const constexpr size_t vect_size = 16 [static], [constexpr], [inherited]
```

### 12.285.3.2 alignment

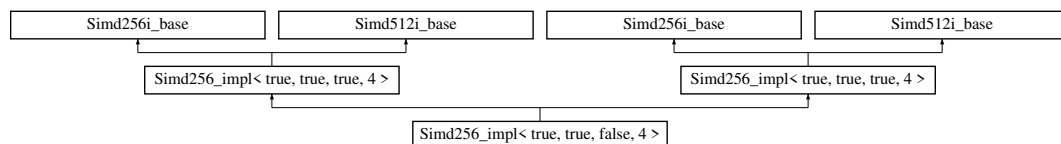
```
const constexpr size_t alignment = 32 [static], [constexpr], [inherited]
```

The documentation for this struct was generated from the following file:

- `simd256 int16.inl`

### 12.286 Simd256\_impl< true, true, false, 4 > Struct Reference

Inheritance diagram for Simd256\_impl< true, true, false, 4 >:



## Data Structures

- union Converter

## Public Types

- using scalar\_t = uint32\_t
- using aligned\_allocator = AlignedAllocator< scalar\_t, Alignment(alignment)>
- using aligned\_vector = std::vector< scalar\_t, aligned\_allocator >

- `template<class Field >`  
`using is_same_element = std::is_same< typename Field::Element, scalar_t >`
- `using simdHalf = Simd128< scalar_t >`
- `using scalar_t = uint32_t`
- `using aligned_allocator = AlignedAllocator< scalar_t, Alignment(alignment)>`
- `using aligned_vector = std::vector< scalar_t, aligned_allocator >`
- `template<class Field >`  
`using is_same_element = std::is_same< typename Field::Element, scalar_t >`
- `using simdHalf = Simd128< scalar_t >`
- `using vect_t = __m256i`
- `using vect_t = __m512i`
- `using half_t = __m128i`
- `using half_t = __m256i`

### Static Public Member Functions

- `static const std::string type_string ()`
- `static INLINE CONST vect_t set1 (const scalar_t x)`
- `static INLINE CONST vect_t set (const scalar_t x0, const scalar_t x1, const scalar_t x2, const scalar_t x3, const scalar_t x4, const scalar_t x5, const scalar_t x6, const scalar_t x7)`
- `template<class T >`  
`static INLINE PURE vect_t gather (const scalar_t *const p, const T *const idx)`
- `static INLINE PURE vect_t load (const scalar_t *const p)`
- `static INLINE PURE vect_t loadu (const scalar_t *const p)`
- `static INLINE void store (scalar_t *p, vect_t v)`
- `static INLINE void storeu (scalar_t *p, vect_t v)`
- `static INLINE void stream (scalar_t *p, const vect_t v)`
- `template<int s>`  
`static INLINE CONST vect_t sra (const vect_t a)`
- `static INLINE CONST vect_t greater (vect_t a, vect_t b)`
- `static INLINE CONST vect_t lesser (vect_t a, vect_t b)`
- `static INLINE CONST vect_t greater_eq (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t lesser_eq (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t mulhi (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t mulx (vect_t a, vect_t b)`
- `static INLINE CONST vect_t fmadddx (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmadddxin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fnmadddx (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fnmadddxin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmsubx (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmsubxin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST scalar_t hadd_to_scal (const vect_t a)`
- `static const std::string type_string ()`
- `static INLINE CONST vect_t set1 (const scalar_t x)`
- `static INLINE CONST vect_t set (const scalar_t x0, const scalar_t x1, const scalar_t x2, const scalar_t x3, const scalar_t x4, const scalar_t x5, const scalar_t x6, const scalar_t x7)`
- `template<class T >`  
`static INLINE PURE vect_t gather (const scalar_t *const p, const T *const idx)`
- `static INLINE PURE vect_t load (const scalar_t *const p)`
- `static INLINE PURE vect_t loadu (const scalar_t *const p)`
- `static INLINE void store (scalar_t *p, vect_t v)`
- `static INLINE void storeu (scalar_t *p, vect_t v)`
- `static INLINE void stream (scalar_t *p, const vect_t v)`
- `template<int s>`  
`static INLINE CONST vect_t sra (const vect_t a)`

- static INLINE CONST vect\_t greater (vect\_t a, vect\_t b)
- static INLINE CONST vect\_t lesser (vect\_t a, vect\_t b)
- static INLINE CONST vect\_t greater\_eq (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t lesser\_eq (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t mulhi (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t mulx (vect\_t a, vect\_t b)
- static INLINE CONST vect\_t fmaddx (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fmaddxin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fnmaddx (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fnmaddxin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fmsubx (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fmsubxin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST scalar\_t hadd\_to\_scal (const vect\_t a)
- template<class T >  
static constexpr bool valid (T \*p)
- template<class T >  
static constexpr bool valid (T \*p)
- template<class T >  
static constexpr bool compliant (T n)
- template<class T >  
static constexpr bool compliant (T n)
- static INLINE CONST vect\_t set1 (const scalar\_t x)
- static INLINE CONST vect\_t set (const scalar\_t x0, const scalar\_t x1, const scalar\_t x2, const scalar\_t x3, const scalar\_t x4, const scalar\_t x5, const scalar\_t x6, const scalar\_t x7)
- static INLINE CONST vect\_t set (const scalar\_t x0, const scalar\_t x1, const scalar\_t x2, const scalar\_t x3, const scalar\_t x4, const scalar\_t x5, const scalar\_t x6, const scalar\_t x7, const scalar\_t x8, const scalar\_t x9, const scalar\_t x10, const scalar\_t x11, const scalar\_t x12, const scalar\_t x13, const scalar\_t x14, const scalar\_t x15)
- template<class T >  
static INLINE PURE vect\_t gather (const scalar\_t \*const p, const T \*const idx)
- static INLINE PURE vect\_t load (const scalar\_t \*const p)
- static INLINE PURE vect\_t loadu (const scalar\_t \*const p)
- static INLINE void store (scalar\_t \*p, vect\_t v)
- static INLINE void storeu (scalar\_t \*p, vect\_t v)
- static INLINE void stream (scalar\_t \*p, const vect\_t v)
- template<int s>  
static INLINE CONST vect\_t sll (const vect\_t a)
- template<int s>  
static INLINE CONST vect\_t sll (const vect\_t a)
- template<int s>  
static INLINE CONST vect\_t srl (const vect\_t a)
- template<int s>  
static INLINE CONST vect\_t srl (const vect\_t a)
- template<uint8\_t s>  
static INLINE CONST vect\_t shuffle\_twice (const vect\_t a)
- template<uint8\_t s>  
static INLINE CONST vect\_t shuffle\_twice (const vect\_t a)
- template<uint32\_t s>  
static INLINE CONST vect\_t shuffle (const vect\_t a)
- template<uint64\_t s>  
static INLINE CONST vect\_t shuffle (const vect\_t a)
- static INLINE CONST vect\_t unpacklo\_intrinsic (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t unpacklo\_intrinsic (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t unpackhi\_intrinsic (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t unpackhi\_intrinsic (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t unpacklo (const vect\_t a, const vect\_t b)

- static INLINE CONST vect\_t unpacklo (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t unpackhi (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t unpackhi (const vect\_t a, const vect\_t b)
- static INLINE void unpacklohi (vect\_t &lo, vect\_t &hi, const vect\_t a, const vect\_t b)
- static INLINE void unpacklohi (vect\_t &lo, vect\_t &hi, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t pack\_even (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t pack\_even (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t pack\_odd (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t pack\_odd (const vect\_t a, const vect\_t b)
- static INLINE void pack (vect\_t &even, vect\_t &odd, const vect\_t a, const vect\_t b)
- static INLINE void pack (vect\_t &even, vect\_t &odd, const vect\_t a, const vect\_t b)
- static INLINE void transpose (vect\_t &r0, vect\_t &r1, vect\_t &r2, vect\_t &r3, vect\_t &r4, vect\_t &r5, vect\_t &r6, vect\_t &r7)
- static INLINE void transpose (vect\_t &r0, vect\_t &r1, vect\_t &r2, vect\_t &r3, vect\_t &r4, vect\_t &r5, vect\_t &r6, vect\_t &r7, vect\_t &r8, vect\_t &r9, vect\_t &r10, vect\_t &r11, vect\_t &r12, vect\_t &r13, vect\_t &r14, vect\_t &r15)
- template<uint8\_t s>  
static INLINE CONST vect\_t blend (const vect\_t a, const vect\_t b)
- template<uint16\_t s>  
static INLINE CONST vect\_t blend (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t add (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t add (const vect\_t a, const vect\_t b)
- static INLINE vect\_t addin (vect\_t &a, const vect\_t b)
- static INLINE vect\_t addin (vect\_t &a, const vect\_t b)
- static INLINE CONST vect\_t sub (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t sub (const vect\_t a, const vect\_t b)
- static INLINE vect\_t subin (vect\_t &a, const vect\_t b)
- static INLINE vect\_t subin (vect\_t &a, const vect\_t b)
- static INLINE CONST vect\_t mullo (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t mullo (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t mul (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t mul (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fmadd (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fmadd (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fmaddin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fmaddin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fnmadd (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fnmadd (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fnmaddin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fnmaddin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fmsub (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fmsub (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fmsubin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fmsubin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t eq (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t eq (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t round (const vect\_t a)
- static INLINE CONST vect\_t round (const vect\_t a)
- static INLINE vect\_t mod (vect\_t &C, const vect\_t &P, const vect\_t &INVP, const vect\_t &NEGP, const vect\_t &MIN, const vect\_t &MAX, vect\_t &Q, vect\_t &T)
- static INLINE vect\_t mod (vect\_t &C, const vect\_t &P, const vect\_t &INVP, const vect\_t &NEGP, const vect\_t &MIN, const vect\_t &MAX, vect\_t &Q, vect\_t &T)
- static INLINE CONST vect\_t zero ()
- static INLINE CONST vect\_t zero ()
- static INLINE CONST vect\_t vor (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t vxor (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t vand (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t vandnot (const vect\_t a, const vect\_t b)



**Static Public Attributes**

- `static const constexpr size_t vect_size = 8`
- `static const constexpr size_t alignment = 32`

**12.286.1 Member Typedef Documentation****12.286.1.1 scalar\_t [1/2]**

```
using scalar_t = uint32_t
```

**12.286.1.2 aligned\_allocator [1/2]**

```
using aligned_allocator = AlignedAllocator<scalar_t, Alignment(alignment)>
```

**12.286.1.3 aligned\_vector [1/2]**

```
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```

**12.286.1.4 is\_same\_element [1/2]**

```
template<class Field >
using is_same_element = std::is_same<typename Field::Element, scalar_t>
```

**12.286.1.5 simdHalf [1/2]**

```
using simdHalf = Simd128<scalar_t>
```

**12.286.1.6 scalar\_t [2/2]**

```
using scalar_t = uint32_t
```

**12.286.1.7 aligned\_allocator [2/2]**

```
using aligned_allocator = AlignedAllocator<scalar_t, Alignment(alignment)>
```

**12.286.1.8 aligned\_vector [2/2]**

```
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```

**12.286.1.9 is\_same\_element [2/2]**

```
template<class Field >
using is_same_element = std::is_same<typename Field::Element, scalar_t>
```

**12.286.1.10 simdHalf [2/2]**

```
using simdHalf = Simd128<scalar_t>
```

**12.286.1.11 vect\_t [1/2]**

```
using vect_t = __m256i [inherited]
```

**12.286.1.12 vect\_t [2/2]**

```
using vect_t = __m512i [inherited]
```

**12.286.1.13 half\_t [1/2]**

```
using half_t = __m128i [inherited]
```

**12.286.1.14 half\_t [2/2]**

```
using half_t = __m256i [inherited]
```

**12.286.2 Member Function Documentation****12.286.2.1 type\_string() [1/2]**

```
static const std::string type_string () [inline], [static]
```

**12.286.2.2 set1() [1/3]**

```
static INLINE CONST vect_t set1 (
 const scalar_t x) [inline], [static]
```

**12.286.2.3 set() [1/4]**

```
static INLINE CONST vect_t set (
 const scalar_t x0,
 const scalar_t x1,
 const scalar_t x2,
 const scalar_t x3,
 const scalar_t x4,
 const scalar_t x5,
 const scalar_t x6,
 const scalar_t x7) [inline], [static]
```

**12.286.2.4 gather() [1/3]**

```
template<class T >
static INLINE PURE vect_t gather (
 const scalar_t *const p,
 const T *const idx) [inline], [static]
```

**12.286.2.5 load() [1/3]**

```
static INLINE PURE vect_t load (
 const scalar_t *const p) [inline], [static]
```

**12.286.2.6 loadu() [1/3]**

```
static INLINE PURE vect_t loadu (
 const scalar_t *const p) [inline], [static]
```

**12.286.2.7 store() [1/3]**

```
static INLINE void store (
 scalar_t * p,
 vect_t v) [inline], [static]
```

**12.286.2.8 storeu() [1/3]**

```
static INLINE void storeu (
 scalar_t * p,
 vect_t v) [inline], [static]
```

**12.286.2.9 stream() [1/3]**

```
static INLINE void stream (
 scalar_t * p,
 const vect_t v) [inline], [static]
```

**12.286.2.10 sra()** [1/2]

```
template<int s>
static INLINE CONST vect_t sra (
 const vect_t a) [inline], [static]
```

**12.286.2.11 greater()** [1/2]

```
static INLINE CONST vect_t greater (
 vect_t a,
 vect_t b) [inline], [static]
```

**12.286.2.12 lesser()** [1/2]

```
static INLINE CONST vect_t lesser (
 vect_t a,
 vect_t b) [inline], [static]
```

**12.286.2.13 greater\_eq()** [1/2]

```
static INLINE CONST vect_t greater_eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.286.2.14 lesser\_eq()** [1/2]

```
static INLINE CONST vect_t lesser_eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.286.2.15 mulhi()** [1/2]

```
static INLINE CONST vect_t mulhi (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.286.2.16 mulx()** [1/2]

```
static INLINE CONST vect_t mulx (
 vect_t a,
 vect_t b) [inline], [static]
```

**12.286.2.17 fmaddx()** [1/2]

```
static INLINE CONST vect_t fmaddx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.286.2.18 fmaddxin()** [1/2]

```
static INLINE vect_t fmaddxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.286.2.19 fmmaddx()** [1/2]

```
static INLINE CONST vect_t fmmaddx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.286.2.20 fmmaddxin()** [1/2]

```
static INLINE vect_t fmmaddxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.286.2.21 fmsubx()** [1/2]

```
static INLINE CONST vect_t fmsubx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.286.2.22 fmsubxin()** [1/2]

```
static INLINE vect_t fmsubxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.286.2.23 hadd\_to\_scal()** [1/2]

```
static INLINE CONST scalar_t hadd_to_scal (
 const vect_t a) [inline], [static]
```

**12.286.2.24 type\_string()** [2/2]

```
static const std::string type_string () [inline], [static]
```

**12.286.2.25 set1()** [2/3]

```
static INLINE CONST vect_t set1 (
 const scalar_t x) [inline], [static]
```

**12.286.2.26 set()** [2/4]

```
static INLINE CONST vect_t set (
 const scalar_t x0,
 const scalar_t x1,
 const scalar_t x2,
 const scalar_t x3,
 const scalar_t x4,
 const scalar_t x5,
 const scalar_t x6,
 const scalar_t x7) [inline], [static]
```

**12.286.2.27 gather()** [2/3]

```
template<class T >
static INLINE PURE vect_t gather (
 const scalar_t *const p,
 const T *const idx) [inline], [static]
```

**12.286.2.28 load()** [2/3]

```
static INLINE PURE vect_t load (
 const scalar_t *const p) [inline], [static]
```

**12.286.2.29 loadu()** [2/3]

```
static INLINE PURE vect_t loadu (
 const scalar_t *const p) [inline], [static]
```

**12.286.2.30 store()** [2/3]

```
static INLINE void store (
 scalar_t * p,
 vect_t v) [inline], [static]
```

**12.286.2.31 storeu()** [2/3]

```
static INLINE void storeu (
 scalar_t * p,
 vect_t v) [inline], [static]
```

**12.286.2.32 stream()** [2/3]

```
static INLINE void stream (
 scalar_t * p,
 const vect_t v) [inline], [static]
```

**12.286.2.33 sra()** [2/2]

```
template<int s>
static INLINE CONST vect_t sra (
 const vect_t a) [inline], [static]
```

**12.286.2.34 greater()** [2/2]

```
static INLINE CONST vect_t greater (
 vect_t a,
 vect_t b) [inline], [static]
```

**12.286.2.35 lesser()** [2/2]

```
static INLINE CONST vect_t lesser (
 vect_t a,
 vect_t b) [inline], [static]
```

**12.286.2.36 greater\_eq()** [2/2]

```
static INLINE CONST vect_t greater_eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.286.2.37 lesser\_eq()** [2/2]

```
static INLINE CONST vect_t lesser_eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.286.2.38 mulhi() [2/2]**

```
static INLINE CONST vect_t mulhi (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.286.2.39 mulx() [2/2]**

```
static INLINE CONST vect_t mulx (
 vect_t a,
 vect_t b) [inline], [static]
```

**12.286.2.40 fmaddx() [2/2]**

```
static INLINE CONST vect_t fmaddx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.286.2.41 fmaddxin() [2/2]**

```
static INLINE vect_t fmaddxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.286.2.42 fnmaddx() [2/2]**

```
static INLINE CONST vect_t fnmaddx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.286.2.43 fnmaddxin() [2/2]**

```
static INLINE vect_t fnmaddxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.286.2.44 fmsubx() [2/2]**

```
static INLINE CONST vect_t fmsubx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.286.2.45 fmsubxin() [2/2]**

```
static INLINE vect_t fmsubxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.286.2.46 hadd\_to\_scal() [2/2]**

```
static INLINE CONST scalar_t hadd_to_scal (
 const vect_t a) [inline], [static]
```

**12.286.2.47 valid() [1/2]**

```
template<class T >
static constexpr bool valid (
 T * p) [inline], [static], [constexpr], [inherited]
```

**12.286.2.48 valid() [2/2]**

```
template<class T >
static constexpr bool valid (
 T * p) [inline], [static], [constexpr], [inherited]
```

**12.286.2.49 compliant() [1/2]**

```
template<class T >
static constexpr bool compliant (
 T n) [inline], [static], [constexpr], [inherited]
```

**12.286.2.50 compliant() [2/2]**

```
template<class T >
static constexpr bool compliant (
 T n) [inline], [static], [constexpr], [inherited]
```

**12.286.2.51 set1() [3/3]**

```
static INLINE CONST vect_t set1 (
 const scalar_t x) [inline], [static], [inherited]
```

**12.286.2.52 set() [3/4]**

```
static INLINE CONST vect_t set (
 const scalar_t x0,
 const scalar_t x1,
 const scalar_t x2,
 const scalar_t x3,
 const scalar_t x4,
 const scalar_t x5,
 const scalar_t x6,
 const scalar_t x7) [inline], [static], [inherited]
```

**12.286.2.53 set() [4/4]**

```
static INLINE CONST vect_t set (
 const scalar_t x0,
 const scalar_t x1,
 const scalar_t x2,
 const scalar_t x3,
 const scalar_t x4,
 const scalar_t x5,
 const scalar_t x6,
 const scalar_t x7,
 const scalar_t x8,
 const scalar_t x9,
 const scalar_t x10,
 const scalar_t x11,
 const scalar_t x12,
 const scalar_t x13,
 const scalar_t x14,
 const scalar_t x15) [inline], [static], [inherited]
```

**12.286.2.54 gather()** [3/3]

```
template<class T >
static INLINE PURE vect_t gather (
 const scalar_t *const p,
 const T *const idx) [inline], [static], [inherited]
```

**12.286.2.55 load()** [3/3]

```
static INLINE PURE vect_t load (
 const scalar_t *const p) [inline], [static], [inherited]
```

**12.286.2.56 loadu()** [3/3]

```
static INLINE PURE vect_t loadu (
 const scalar_t *const p) [inline], [static], [inherited]
```

**12.286.2.57 store()** [3/3]

```
static INLINE void store (
 scalar_t * p,
 vect_t v) [inline], [static], [inherited]
```

**12.286.2.58 storeu()** [3/3]

```
static INLINE void storeu (
 scalar_t * p,
 vect_t v) [inline], [static], [inherited]
```

**12.286.2.59 stream()** [3/3]

```
static INLINE void stream (
 scalar_t * p,
 const vect_t v) [inline], [static], [inherited]
```

**12.286.2.60 sll()** [1/2]

```
template<int s>
static INLINE CONST vect_t sll (
 const vect_t a) [inline], [static], [inherited]
```

**12.286.2.61 srl()** [2/2]

```
template<int s>
static INLINE CONST vect_t srl (
 const vect_t a) [inline], [static], [inherited]
```

**12.286.2.62 srl()** [1/2]

```
template<int s>
static INLINE CONST vect_t srl (
 const vect_t a) [inline], [static], [inherited]
```

**12.286.2.63 srl()** [2/2]

```
template<int s>
static INLINE CONST vect_t srl (
 const vect_t a) [inline], [static], [inherited]
```



**12.286.2.64 shuffle\_twice() [1/2]**

```
template<uint8_t s>
static INLINE CONST vect_t shuffle_twice (
 const vect_t a) [inline], [static], [inherited]
```

**12.286.2.65 shuffle\_twice() [2/2]**

```
template<uint8_t s>
static INLINE CONST vect_t shuffle_twice (
 const vect_t a) [inline], [static], [inherited]
```

**12.286.2.66 shuffle() [1/2]**

```
template<uint32_t s>
static INLINE CONST vect_t shuffle (
 const vect_t a) [inline], [static], [inherited]
```

**12.286.2.67 shuffle() [2/2]**

```
template<uint64_t s>
static INLINE CONST vect_t shuffle (
 const vect_t a) [inline], [static], [inherited]
```

**12.286.2.68 unpacklo\_intrinsic() [1/2]**

```
static INLINE CONST vect_t unpacklo_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.69 unpacklo\_intrinsic() [2/2]**

```
static INLINE CONST vect_t unpacklo_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.70 unpackhi\_intrinsic() [1/2]**

```
static INLINE CONST vect_t unpackhi_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.71 unpackhi\_intrinsic() [2/2]**

```
static INLINE CONST vect_t unpackhi_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.72 unpacklo() [1/2]**

```
static INLINE CONST vect_t unpacklo (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.73 unpacklo() [2/2]**

```
static INLINE CONST vect_t unpacklo (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.74 unpackhi() [1/2]**

```
static INLINE CONST vect_t unpackhi (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.75 unpackhi() [2/2]**

```
static INLINE CONST vect_t unpackhi (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.76 unpacklohi() [1/2]**

```
static INLINE void unpacklohi (
 vect_t & lo,
 vect_t & hi,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.77 unpacklohi() [2/2]**

```
static INLINE void unpacklohi (
 vect_t & lo,
 vect_t & hi,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.78 pack\_even() [1/2]**

```
static INLINE CONST vect_t pack_even (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.79 pack\_even() [2/2]**

```
static INLINE CONST vect_t pack_even (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.80 pack\_odd() [1/2]**

```
static INLINE CONST vect_t pack_odd (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.81 pack\_odd() [2/2]**

```
static INLINE CONST vect_t pack_odd (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.82 pack() [1/2]**

```
static INLINE void pack (
 vect_t & even,
 vect_t & odd,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.83 pack()** [2/2]

```
static INLINE void pack (
 vect_t & even,
 vect_t & odd,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.84 transpose()** [1/2]

```
static INLINE void transpose (
 vect_t & r0,
 vect_t & r1,
 vect_t & r2,
 vect_t & r3,
 vect_t & r4,
 vect_t & r5,
 vect_t & r6,
 vect_t & r7) [inline], [static], [inherited]
```

**12.286.2.85 transpose()** [2/2]

```
static INLINE void transpose (
 vect_t & r0,
 vect_t & r1,
 vect_t & r2,
 vect_t & r3,
 vect_t & r4,
 vect_t & r5,
 vect_t & r6,
 vect_t & r7,
 vect_t & r8,
 vect_t & r9,
 vect_t & r10,
 vect_t & r11,
 vect_t & r12,
 vect_t & r13,
 vect_t & r14,
 vect_t & r15) [inline], [static], [inherited]
```

**12.286.2.86 blend()** [1/2]

```
template<uint8_t s>
static INLINE CONST vect_t blend (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.87 blend()** [2/2]

```
template<uint16_t s>
static INLINE CONST vect_t blend (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.88 add()** [1/2]

```
static INLINE CONST vect_t add (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.89 add()** [2/2]

```
static INLINE CONST vect_t add (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.90 addin()** [1/2]

```
static INLINE vect_t addin (
 vect_t & a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.91 addin()** [2/2]

```
static INLINE vect_t addin (
 vect_t & a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.92 sub()** [1/2]

```
static INLINE CONST vect_t sub (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.93 sub()** [2/2]

```
static INLINE CONST vect_t sub (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.94 subin()** [1/2]

```
static INLINE vect_t subin (
 vect_t & a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.95 subin()** [2/2]

```
static INLINE vect_t subin (
 vect_t & a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.96 mullo()** [1/2]

```
static INLINE CONST vect_t mullo (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.97 mullo()** [2/2]

```
static INLINE CONST vect_t mullo (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.98 mul()** [1/2]

```
static INLINE CONST vect_t mul (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.99 mul()** [2/2]

```
static INLINE CONST vect_t mul (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.100 fmadd()** [1/2]

```
static INLINE CONST vect_t fmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.101 fmadd()** [2/2]

```
static INLINE CONST vect_t fmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.102 fmaddin()** [1/2]

```
static INLINE vect_t fmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.103 fmaddin()** [2/2]

```
static INLINE vect_t fmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.104 fnmadd()** [1/2]

```
static INLINE CONST vect_t fnmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.105 fnmadd()** [2/2]

```
static INLINE CONST vect_t fnmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.106 fnmaddin()** [1/2]

```
static INLINE vect_t fnmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.286.2.107 fnmaddin()** [2/2]

```
static INLINE vect_t fnmaddin (
 vect_t & c,
```

```

 const vect_t a,
 const vect_t b) [inline], [static], [inherited]

```

#### 12.286.2.108 fmsub() [1/2]

```

static INLINE CONST vect_t fmsub (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]

```

#### 12.286.2.109 fmsub() [2/2]

```

static INLINE CONST vect_t fmsub (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]

```

#### 12.286.2.110 fmsubin() [1/2]

```

static INLINE vect_t fmsubin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]

```

#### 12.286.2.111 fmsubin() [2/2]

```

static INLINE vect_t fmsubin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]

```

#### 12.286.2.112 eq() [1/2]

```

static INLINE CONST vect_t eq (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]

```

#### 12.286.2.113 eq() [2/2]

```

static INLINE CONST vect_t eq (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]

```

#### 12.286.2.114 round() [1/2]

```

static INLINE CONST vect_t round (
 const vect_t a) [inline], [static], [inherited]

```

#### 12.286.2.115 round() [2/2]

```

static INLINE CONST vect_t round (
 const vect_t a) [inline], [static], [inherited]

```

#### 12.286.2.116 mod() [1/2]

```

static INLINE vect_t mod (
 vect_t & C,
 const vect_t & P,
 const vect_t & INVP,
 const vect_t & NEGP,

```

```

const vect_t & MIN,
const vect_t & MAX,
vect_t & Q,
vect_t & T) [inline], [static], [inherited]

```

**12.286.2.117 mod() [2/2]**

```

static INLINE vect_t mod (
 vect_t & C,
 const vect_t & P,
 const vect_t & INVP,
 const vect_t & NEGP,
 const vect_t & MIN,
 const vect_t & MAX,
 vect_t & Q,
 vect_t & T) [inline], [static], [inherited]

```

**12.286.2.118 zero() [1/2]**

```

static INLINE CONST vect_t zero () [inline], [static], [inherited]

```

**12.286.2.119 zero() [2/2]**

```

static INLINE CONST vect_t zero () [inline], [static], [inherited]

```

**12.286.2.120 vor()**

```

static INLINE CONST vect_t vor (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]

```

**12.286.2.121 vxor()**

```

static INLINE CONST vect_t vxor (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]

```

**12.286.2.122 vand()**

```

static INLINE CONST vect_t vand (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]

```

**12.286.2.123 vandnot()**

```

static INLINE CONST vect_t vandnot (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]

```

**12.286.3 Field Documentation****12.286.3.1 vect\_size**

```

static const constexpr size_t vect_size = 8 [static], [constexpr], [inherited]

```

**12.286.3.2 alignment**

```

static const constexpr size_t alignment = 32 [static], [constexpr], [inherited]

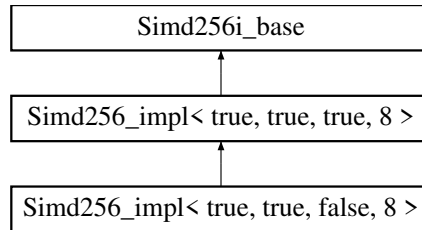
```

The documentation for this struct was generated from the following files:

- [simd256\\_int32.inl](#)
- [simd512\\_int32.inl](#)

## 12.287 Simd256\_impl< true, true, false, 8 > Struct Reference

Inheritance diagram for Simd256\_impl< true, true, false, 8 >:



### Data Structures

- union [Converter](#)

### Public Types

- using `scalar_t` = `uint64_t`
- using `aligned_allocator` = `AlignedAllocator< scalar_t, Alignment(alignment)>`
- using `aligned_vector` = `std::vector< scalar_t, aligned_allocator >`
- template<class `Field` >  
using `is_same_element` = `std::is_same< typename Field::Element, scalar_t >`
- using `simdHalf` = `Simd128< scalar_t >`
- using `vect_t` = `__m256i`
- using `half_t` = `__m128i`

### Static Public Member Functions

- static const `std::string type_string()`
- static `INLINE CONST vect_t set1 (const scalar_t x)`
- static `INLINE CONST vect_t set (const scalar_t x0, const scalar_t x1, const scalar_t x2, const scalar_t x3)`
- template<class `T` >  
static `INLINE PURE vect_t gather (const scalar_t *const p, const T *const idx)`
- static `INLINE PURE vect_t load (const scalar_t *const p)`
- static `INLINE PURE vect_t loadu (const scalar_t *const p)`
- static `INLINE void store (scalar_t *p, vect_t v)`
- static `INLINE void storeu (scalar_t *p, vect_t v)`
- static `INLINE void stream (scalar_t *p, const vect_t v)`
- template<int `s`>  
static `INLINE CONST vect_t sra (const vect_t a)`
- static `INLINE CONST vect_t greater (vect_t a, vect_t b)`
- static `INLINE CONST vect_t lesser (vect_t a, vect_t b)`
- static `INLINE CONST vect_t greater_eq (const vect_t a, const vect_t b)`
- static `INLINE CONST vect_t lesser_eq (const vect_t a, const vect_t b)`
- static `INLINE CONST vect_t mullo (vect_t a, vect_t b)`
- static `INLINE CONST vect_t mulhi (vect_t a, vect_t b)`
- static `INLINE CONST vect_t mulx (const vect_t a, const vect_t b)`
- static `INLINE CONST vect_t fmaddx (const vect_t c, const vect_t a, const vect_t b)`
- static `INLINE vect_t fmaddxin (vect_t &c, const vect_t a, const vect_t b)`
- static `INLINE CONST vect_t fnmaddx (const vect_t c, const vect_t a, const vect_t b)`
- static `INLINE vect_t fnmaddxin (vect_t &c, const vect_t a, const vect_t b)`



- static INLINE CONST vect\_t fmsubx (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fmsubxin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST scalar\_t hadd\_to\_scal (const vect\_t a)
- template<class T >  
static constexpr bool valid (T \*p)
- template<class T >  
static constexpr bool compliant (T n)
- static INLINE CONST vect\_t set1 (const scalar\_t x)
- static INLINE CONST vect\_t set (const scalar\_t x0, const scalar\_t x1, const scalar\_t x2, const scalar\_t x3)
- template<class T >  
static INLINE PURE vect\_t gather (const scalar\_t \*const p, const T \*const idx)
- template<int idx>  
static INLINE CONST scalar\_t get (vect\_t v)
- static INLINE PURE vect\_t load (const scalar\_t \*const p)
- static INLINE PURE vect\_t loadu (const scalar\_t \*const p)
- static INLINE void store (scalar\_t \*p, vect\_t v)
- static INLINE void storeu (scalar\_t \*p, vect\_t v)
- static INLINE void stream (scalar\_t \*p, const vect\_t v)
- template<int s>  
static INLINE CONST vect\_t sll (const vect\_t a)
- template<int s>  
static INLINE CONST vect\_t srl (const vect\_t a)
- template<uint8\_t s>  
static INLINE CONST vect\_t shuffle (const vect\_t a)
- static INLINE CONST vect\_t unpacklo\_intrinsic (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t unpackhi\_intrinsic (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t unpacklo (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t unpackhi (const vect\_t a, const vect\_t b)
- static INLINE void unpacklohi (vect\_t &lo, vect\_t &hi, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t pack\_even (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t pack\_odd (const vect\_t a, const vect\_t b)
- static INLINE void pack (vect\_t &even, vect\_t &odd, const vect\_t a, const vect\_t b)
- static INLINE void transpose (vect\_t &r0, vect\_t &r1, vect\_t &r2, vect\_t &r3)
- template<uint8\_t s>  
static INLINE CONST vect\_t blend (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t add (const vect\_t a, const vect\_t b)
- static INLINE vect\_t addin (vect\_t &a, const vect\_t b)
- static INLINE CONST vect\_t sub (const vect\_t a, const vect\_t b)
- static INLINE vect\_t subin (vect\_t &a, const vect\_t b)
- static INLINE CONST vect\_t mul (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fmadd (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fmaddin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fnmadd (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fnmaddin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fmsub (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fmsubin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t eq (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t round (const vect\_t a)
- static INLINE CONST vect\_t mask\_high ()
- static INLINE CONST vect\_t mulhi\_fast (vect\_t x, vect\_t y)
- static INLINE vect\_t mod (vect\_t &C, const \_\_m256d &P, const \_\_m256d &INVP, const \_\_m256d &NEGP, const vect\_t &POW50REM, const \_\_m256d &MIN, const \_\_m256d &MAX, \_\_m256d &Q, \_\_m256d &T)
- static INLINE CONST vect\_t zero ()

**Static Public Attributes**

- `static const constexpr size_t vect_size = 4`
- `static const constexpr size_t alignment = 32`

**Static Protected Member Functions**

- `static INLINE CONST vect_t signbits (const vect_t x)`

**12.287.1 Member Typedef Documentation****12.287.1.1 scalar\_t**

```
using scalar_t = uint64_t
```

**12.287.1.2 aligned\_allocator**

```
using aligned_allocator = AlignedAllocator<scalar_t, Alignment(alignment)>
```

**12.287.1.3 aligned\_vector**

```
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```

**12.287.1.4 is\_same\_element**

```
template<class Field >
using is_same_element = std::is_same<typename Field::Element, scalar_t>
```

**12.287.1.5 simdHalf**

```
using simdHalf = Simd128<scalar_t>
```

**12.287.1.6 vect\_t**

```
using vect_t = __m256i [inherited]
```

**12.287.1.7 half\_t**

```
using half_t = __m128i [inherited]
```

**12.287.2 Member Function Documentation****12.287.2.1 type\_string()**

```
static const std::string type_string () [inline], [static]
```

**12.287.2.2 set1() [1/2]**

```
static INLINE CONST vect_t set1 (
 const scalar_t x) [inline], [static]
```

**12.287.2.3 set() [1/2]**

```
static INLINE CONST vect_t set (
 const scalar_t x0,
 const scalar_t x1,
 const scalar_t x2,
 const scalar_t x3) [inline], [static]
```

**12.287.2.4 gather()** [1/2]

```
template<class T >
static INLINE PURE vect_t gather (
 const scalar_t *const p,
 const T *const idx) [inline], [static]
```

**12.287.2.5 load()** [1/2]

```
static INLINE PURE vect_t load (
 const scalar_t *const p) [inline], [static]
```

**12.287.2.6 loadu()** [1/2]

```
static INLINE PURE vect_t loadu (
 const scalar_t *const p) [inline], [static]
```

**12.287.2.7 store()** [1/2]

```
static INLINE void store (
 scalar_t * p,
 vect_t v) [inline], [static]
```

**12.287.2.8 storeu()** [1/2]

```
static INLINE void storeu (
 scalar_t * p,
 vect_t v) [inline], [static]
```

**12.287.2.9 stream()** [1/2]

```
static INLINE void stream (
 scalar_t * p,
 const vect_t v) [inline], [static]
```

**12.287.2.10 sra()**

```
template<int s>
static INLINE CONST vect_t sra (
 const vect_t a) [inline], [static]
```

**12.287.2.11 greater()**

```
static INLINE CONST vect_t greater (
 vect_t a,
 vect_t b) [inline], [static]
```

**12.287.2.12 lesser()**

```
static INLINE CONST vect_t lesser (
 vect_t a,
 vect_t b) [inline], [static]
```

**12.287.2.13 greater\_eq()**

```
static INLINE CONST vect_t greater_eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.287.2.14 lesser\_eq()**

```
static INLINE CONST vect_t lesser_eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.287.2.15 mullo()**

```
static INLINE CONST vect_t mullo (
 vect_t a,
 vect_t b) [inline], [static]
```

**12.287.2.16 mulhi()**

```
static INLINE CONST vect_t mulhi (
 vect_t a,
 vect_t b) [inline], [static]
```

**12.287.2.17 mulx()**

```
static INLINE CONST vect_t mulx (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.287.2.18 fmaddx()**

```
static INLINE CONST vect_t fmaddx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.287.2.19 fmaddxin()**

```
static INLINE vect_t fmaddxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.287.2.20 fnmaddx()**

```
static INLINE CONST vect_t fnmaddx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.287.2.21 fnmaddxin()**

```
static INLINE vect_t fnmaddxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.287.2.22 fmsubx()**

```
static INLINE CONST vect_t fmsubx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.287.2.23 fmsubxin()**

```
static INLINE vect_t fmsubxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.287.2.24 hadd\_to\_scal()**

```
static INLINE CONST scalar_t hadd_to_scal (
 const vect_t a) [inline], [static]
```

**12.287.2.25 valid()**

```
template<class T >
static constexpr bool valid (
 T * p) [inline], [static], [constexpr], [inherited]
```

**12.287.2.26 compliant()**

```
template<class T >
static constexpr bool compliant (
 T n) [inline], [static], [constexpr], [inherited]
```

**12.287.2.27 set1() [2/2]**

```
static INLINE CONST vect_t set1 (
 const scalar_t x) [inline], [static], [inherited]
```

**12.287.2.28 set() [2/2]**

```
static INLINE CONST vect_t set (
 const scalar_t x0,
 const scalar_t x1,
 const scalar_t x2,
 const scalar_t x3) [inline], [static], [inherited]
```

**12.287.2.29 gather() [2/2]**

```
template<class T >
static INLINE PURE vect_t gather (
 const scalar_t *const p,
 const T *const idx) [inline], [static], [inherited]
```

**12.287.2.30 get()**

```
template<int idx>
static INLINE CONST scalar_t get (
 vect_t v) [inline], [static], [inherited]
```

**12.287.2.31 load() [2/2]**

```
static INLINE PURE vect_t load (
 const scalar_t *const p) [inline], [static], [inherited]
```

**12.287.2.32 loadu() [2/2]**

```
static INLINE PURE vect_t loadu (
 const scalar_t *const p) [inline], [static], [inherited]
```

**12.287.2.33 store()** [2/2]

```
static INLINE void store (
 scalar_t * p,
 vect_t v) [inline], [static], [inherited]
```

**12.287.2.34 storeu()** [2/2]

```
static INLINE void storeu (
 scalar_t * p,
 vect_t v) [inline], [static], [inherited]
```

**12.287.2.35 stream()** [2/2]

```
static INLINE void stream (
 scalar_t * p,
 const vect_t v) [inline], [static], [inherited]
```

**12.287.2.36 sll()**

```
template<int s>
static INLINE CONST vect_t sll (
 const vect_t a) [inline], [static], [inherited]
```

**12.287.2.37 srl()**

```
template<int s>
static INLINE CONST vect_t srl (
 const vect_t a) [inline], [static], [inherited]
```

**12.287.2.38 shuffle()**

```
template<uint8_t s>
static INLINE CONST vect_t shuffle (
 const vect_t a) [inline], [static], [inherited]
```

**12.287.2.39 unpacklo\_intrinsic()**

```
static INLINE CONST vect_t unpacklo_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.287.2.40 unpackhi\_intrinsic()**

```
static INLINE CONST vect_t unpackhi_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.287.2.41 unpacklo()**

```
static INLINE CONST vect_t unpacklo (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.287.2.42 unpackhi()**

```
static INLINE CONST vect_t unpackhi (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.287.2.43 unpacklohi()**

```
static INLINE void unpacklohi (
 vect_t & lo,
 vect_t & hi,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.287.2.44 pack\_even()**

```
static INLINE CONST vect_t pack_even (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.287.2.45 pack\_odd()**

```
static INLINE CONST vect_t pack_odd (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.287.2.46 pack()**

```
static INLINE void pack (
 vect_t & even,
 vect_t & odd,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.287.2.47 transpose()**

```
static INLINE void transpose (
 vect_t & r0,
 vect_t & r1,
 vect_t & r2,
 vect_t & r3) [inline], [static], [inherited]
```

**12.287.2.48 blend()**

```
template<uint8_t s>
static INLINE CONST vect_t blend (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.287.2.49 add()**

```
static INLINE CONST vect_t add (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.287.2.50 addin()**

```
static INLINE vect_t addin (
 vect_t & a,
 const vect_t b) [inline], [static], [inherited]
```

**12.287.2.51 sub()**

```
static INLINE CONST vect_t sub (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.287.2.52 subin()**

```
static INLINE vect_t subin (
 vect_t & a,
 const vect_t b) [inline], [static], [inherited]
```

**12.287.2.53 mul()**

```
static INLINE CONST vect_t mul (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.287.2.54 fmadd()**

```
static INLINE CONST vect_t fmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.287.2.55 fmaddin()**

```
static INLINE vect_t fmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.287.2.56 fnmadd()**

```
static INLINE CONST vect_t fnmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.287.2.57 fnmaddin()**

```
static INLINE vect_t fnmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.287.2.58 fmsub()**

```
static INLINE CONST vect_t fmsub (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.287.2.59 fmsubin()**

```
static INLINE vect_t fmsubin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.287.2.60 eq()**

```
static INLINE CONST vect_t eq (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```



**12.287.2.61 round()**

```
static INLINE CONST vect_t round (
 const vect_t a) [inline], [static], [inherited]
```

**12.287.2.62 mask\_high()**

```
static INLINE CONST vect_t mask_high () [inline], [static], [inherited]
```

**12.287.2.63 mulhi\_fast()**

```
INLINE CONST vect_t mulhi_fast (
 vect_t x,
 vect_t y) [static], [inherited]
```

**12.287.2.64 mod()**

```
INLINE vect_t mod (
 vect_t & C,
 const __m256d & P,
 const __m256d & INV_P,
 const __m256d & NEGP,
 const vect_t & POW50REM,
 const __m256d & MIN,
 const __m256d & MAX,
 __m256d & Q,
 __m256d & T) [static], [inherited]
```

**12.287.2.65 signbits()**

```
static INLINE CONST vect_t signbits (
 const vect_t x) [inline], [static], [protected], [inherited]
```

**12.287.2.66 zero()**

```
static INLINE CONST vect_t zero () [inline], [static], [inherited]
```

**12.287.3 Field Documentation****12.287.3.1 vect\_size**

```
const constexpr size_t vect_size = 4 [static], [constexpr], [inherited]
```

**12.287.3.2 alignment**

```
const constexpr size_t alignment = 32 [static], [constexpr], [inherited]
```

The documentation for this struct was generated from the following file:

- [simd256\\_int64.inl](#)

**12.288 Simd256\_impl< true, true, true, 2 > Struct Reference**

Inheritance diagram for Simd256\_impl< true, true, true, 2 >:



## Data Structures

- union [Converter](#)

## Public Types

- using vect\_t = \_\_m256i
- using half\_t = \_\_m128i
- using scalar\_t = int16\_t
- using simdHalf = Simd128< scalar\_t >
- using aligned\_allocator = AlignedAllocator< scalar\_t, Alignment(alignment)>
- using aligned\_vector = std::vector< scalar\_t, aligned\_allocator >
- template<class Field >  
using is\_same\_element = std::is\_same< typename Field::Element, scalar\_t >

## Static Public Member Functions

- static const std::string type\_string ()
- template<class T >  
static constexpr bool valid (T \*p)
- template<class T >  
static constexpr bool compliant (T n)
- static INLINE CONST vect\_t set1 (const scalar\_t x)
- static INLINE CONST vect\_t set (const scalar\_t x0, const scalar\_t x1, const scalar\_t x2, const scalar\_t x3, const scalar\_t x4, const scalar\_t x5, const scalar\_t x6, const scalar\_t x7, const scalar\_t x8, const scalar\_t x9, const scalar\_t x10, const scalar\_t x11, const scalar\_t x12, const scalar\_t x13, const scalar\_t x14, const scalar\_t x15)
- template<class T >  
static INLINE PURE vect\_t gather (const scalar\_t \*const p, const T \*const idx)
- static INLINE PURE vect\_t load (const scalar\_t \*const p)
- static INLINE PURE vect\_t loadu (const scalar\_t \*const p)
- static INLINE void store (scalar\_t \*p, vect\_t v)
- static INLINE void storeu (scalar\_t \*p, vect\_t v)
- static INLINE void stream (scalar\_t \*p, const vect\_t v)
- template<int s>  
static INLINE CONST vect\_t sll (const vect\_t a)
- template<int s>  
static INLINE CONST vect\_t srl (const vect\_t a)
- template<int s>  
static INLINE CONST vect\_t sra (const vect\_t a)
- template<uint64\_t s>  
static INLINE CONST vect\_t shuffle (const vect\_t a)
- static INLINE CONST vect\_t unpacklo\_intrinsic (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t unpackhi\_intrinsic (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t unpacklo (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t unpackhi (const vect\_t a, const vect\_t b)
- static INLINE void unpacklohi (vect\_t &lo, vect\_t &hi, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t pack\_even (const vect\_t a, const vect\_t b)

- `static INLINE CONST vect_t pack_odd (const vect_t a, const vect_t b)`
- `static INLINE void pack (vect_t &even, vect_t &odd, const vect_t a, const vect_t b)`
- `static INLINE void transpose (vect_t &r0, vect_t &r1, vect_t &r2, vect_t &r3, vect_t &r4, vect_t &r5, vect_t &r6, vect_t &r7, vect_t &r8, vect_t &r9, vect_t &r10, vect_t &r11, vect_t &r12, vect_t &r13, vect_t &r14, vect_t &r15)`
- `template<uint16_t s, typename std::enable_if<(s & 0x00ff) == (s > 8)>::type * = nullptr>`  
`static INLINE vect_t blend (const vect_t a, const vect_t b)`
- `template<uint16_t s, typename std::enable_if<(s & 0x00ff) != (s > 8)>::type * = nullptr>`  
`static INLINE vect_t blend (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t add (const vect_t a, const vect_t b)`
- `static INLINE vect_t addin (vect_t &a, const vect_t b)`
- `static INLINE CONST vect_t sub (const vect_t a, const vect_t b)`
- `static INLINE vect_t subin (vect_t &a, const vect_t b)`
- `static INLINE CONST vect_t mullo (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t mul (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t mulhi (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t mulx (vect_t a, vect_t b)`
- `static INLINE CONST vect_t fmadd (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmaddin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmaddx (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmaddxin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fnmadd (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fnmaddin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fnmaddx (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fnmaddxin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmsub (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmsubin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmsubx (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmsubxin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t eq (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t greater (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t lesser (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t greater_eq (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t lesser_eq (const vect_t a, const vect_t b)`
- `static INLINE CONST scalar_t hadd_to_scal (const vect_t a)`
- `static INLINE CONST vect_t round (const vect_t a)`
- `static INLINE vect_t mod (vect_t &C, const vect_t &P, const vect_t &INVP, const vect_t &NEGP, const vect_t &MIN, const vect_t &MAX, vect_t &Q, vect_t &T)`
- `static INLINE CONST vect_t zero ()`

### Static Public Attributes

- `static const constexpr size_t vect_size = 16`
- `static const constexpr size_t alignment = 32`

## 12.288.1 Member Typedef Documentation

### 12.288.1.1 vect\_t

```
using vect_t = __m256i
```

### 12.288.1.2 half\_t

```
using half_t = __m128i
```

### 12.288.1.3 scalar\_t

```
using scalar_t = int16_t
```

**12.288.1.4 simdHalf**

```
using simdHalf = Simd128<scalar_t>
```

**12.288.1.5 aligned\_allocator**

```
using aligned_allocator = AlignedAllocator<scalar_t, Alignment(alignment)>
```

**12.288.1.6 aligned\_vector**

```
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```

**12.288.1.7 is\_same\_element**

```
template<class Field >
using is_same_element = std::is_same<typename Field::Element, scalar_t>
```

**12.288.2 Member Function Documentation****12.288.2.1 type\_string()**

```
static const std::string type_string () [inline], [static]
```

**12.288.2.2 valid()**

```
template<class T >
static constexpr bool valid (
 T * p) [inline], [static], [constexpr]
```

**12.288.2.3 compliant()**

```
template<class T >
static constexpr bool compliant (
 T n) [inline], [static], [constexpr]
```

**12.288.2.4 set1()**

```
static INLINE CONST vect_t set1 (
 const scalar_t x) [inline], [static]
```

**12.288.2.5 set()**

```
static INLINE CONST vect_t set (
 const scalar_t x0,
 const scalar_t x1,
 const scalar_t x2,
 const scalar_t x3,
 const scalar_t x4,
 const scalar_t x5,
 const scalar_t x6,
 const scalar_t x7,
 const scalar_t x8,
 const scalar_t x9,
 const scalar_t x10,
 const scalar_t x11,
 const scalar_t x12,
 const scalar_t x13,
 const scalar_t x14,
 const scalar_t x15) [inline], [static]
```

### 12.288.2.6 gather()

```
template<class T >
static INLINE PURE vect_t gather (
 const scalar_t *const p,
 const T *const idx) [inline], [static]
```

### 12.288.2.7 load()

```
static INLINE PURE vect_t load (
 const scalar_t *const p) [inline], [static]
```

### 12.288.2.8 loadu()

```
static INLINE PURE vect_t loadu (
 const scalar_t *const p) [inline], [static]
```

### 12.288.2.9 store()

```
static INLINE void store (
 scalar_t * p,
 vect_t v) [inline], [static]
```

### 12.288.2.10 storeu()

```
static INLINE void storeu (
 scalar_t * p,
 vect_t v) [inline], [static]
```

### 12.288.2.11 stream()

```
static INLINE void stream (
 scalar_t * p,
 const vect_t v) [inline], [static]
```

### 12.288.2.12 sll()

```
template<int s>
static INLINE CONST vect_t sll (
 const vect_t a) [inline], [static]
```

### 12.288.2.13 srl()

```
template<int s>
static INLINE CONST vect_t srl (
 const vect_t a) [inline], [static]
```

### 12.288.2.14 sra()

```
template<int s>
static INLINE CONST vect_t sra (
 const vect_t a) [inline], [static]
```

### 12.288.2.15 shuffle()

```
template<uint64_t s>
static INLINE CONST vect_t shuffle (
 const vect_t a) [inline], [static]
```

**12.288.2.16 unpacklo\_intrinsic()**

```
static INLINE CONST vect_t unpacklo_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.288.2.17 unpackhi\_intrinsic()**

```
static INLINE CONST vect_t unpackhi_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.288.2.18 unpacklo()**

```
static INLINE CONST vect_t unpacklo (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.288.2.19 unpackhi()**

```
static INLINE CONST vect_t unpackhi (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.288.2.20 unpacklohi()**

```
static INLINE void unpacklohi (
 vect_t & lo,
 vect_t & hi,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.288.2.21 pack\_even()**

```
static INLINE CONST vect_t pack_even (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.288.2.22 pack\_odd()**

```
static INLINE CONST vect_t pack_odd (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.288.2.23 pack()**

```
static INLINE void pack (
 vect_t & even,
 vect_t & odd,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.288.2.24 transpose()**

```
static INLINE void transpose (
 vect_t & r0,
 vect_t & r1,
 vect_t & r2,
 vect_t & r3,
 vect_t & r4,
```

```

 vect_t & r5,
 vect_t & r6,
 vect_t & r7,
 vect_t & r8,
 vect_t & r9,
 vect_t & r10,
 vect_t & r11,
 vect_t & r12,
 vect_t & r13,
 vect_t & r14,
 vect_t & r15) [inline], [static]

```

**12.288.2.25 blend() [1/2]**

```

template<uint16_t s, typename std::enable_if<(s &0x00ff)==(s > > 8)>::type * = nullptr>
static INLINE vect_t blend (
 const vect_t a,
 const vect_t b) [inline], [static]

```

**12.288.2.26 blend() [2/2]**

```

template<uint16_t s, typename std::enable_if<(s &0x00ff) !=(s > > 8)>::type * = nullptr>
static INLINE vect_t blend (
 const vect_t a,
 const vect_t b) [inline], [static]

```

**12.288.2.27 add()**

```

static INLINE CONST vect_t add (
 const vect_t a,
 const vect_t b) [inline], [static]

```

**12.288.2.28 addin()**

```

static INLINE vect_t addin (
 vect_t & a,
 const vect_t b) [inline], [static]

```

**12.288.2.29 sub()**

```

static INLINE CONST vect_t sub (
 const vect_t a,
 const vect_t b) [inline], [static]

```

**12.288.2.30 subin()**

```

static INLINE vect_t subin (
 vect_t & a,
 const vect_t b) [inline], [static]

```

**12.288.2.31 mullo()**

```

static INLINE CONST vect_t mullo (
 const vect_t a,
 const vect_t b) [inline], [static]

```

**12.288.2.32 mul()**

```
static INLINE CONST vect_t mul (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.288.2.33 mulhi()**

```
static INLINE CONST vect_t mulhi (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.288.2.34 mulx()**

```
static INLINE CONST vect_t mulx (
 vect_t a,
 vect_t b) [inline], [static]
```

**12.288.2.35 fmadd()**

```
static INLINE CONST vect_t fmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.288.2.36 fmaddin()**

```
static INLINE vect_t fmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.288.2.37 fmaddx()**

```
static INLINE CONST vect_t fmaddx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.288.2.38 fmaddxin()**

```
static INLINE vect_t fmaddxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.288.2.39 fnmadd()**

```
static INLINE CONST vect_t fnmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.288.2.40 fnmaddin()**

```
static INLINE vect_t fnmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```



**12.288.2.41 fnmaddx()**

```
static INLINE CONST vect_t fnmaddx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.288.2.42 fnmaddxin()**

```
static INLINE vect_t fnmaddxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.288.2.43 fmsub()**

```
static INLINE CONST vect_t fmsub (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.288.2.44 fmsubin()**

```
static INLINE vect_t fmsubin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.288.2.45 fmsubx()**

```
static INLINE CONST vect_t fmsubx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.288.2.46 fmsubxin()**

```
static INLINE vect_t fmsubxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.288.2.47 eq()**

```
static INLINE CONST vect_t eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.288.2.48 greater()**

```
static INLINE CONST vect_t greater (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.288.2.49 lesser()**

```
static INLINE CONST vect_t lesser (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.288.2.50 greater\_eq()**

```
static INLINE CONST vect_t greater_eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.288.2.51 lesser\_eq()**

```
static INLINE CONST vect_t lesser_eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.288.2.52 hadd\_to\_scal()**

```
static INLINE CONST scalar_t hadd_to_scal (
 const vect_t a) [inline], [static]
```

**12.288.2.53 round()**

```
static INLINE CONST vect_t round (
 const vect_t a) [inline], [static]
```

**12.288.2.54 mod()**

```
static INLINE vect_t mod (
 vect_t & C,
 const vect_t & P,
 const vect_t & INVP,
 const vect_t & NEGP,
 const vect_t & MIN,
 const vect_t & MAX,
 vect_t & Q,
 vect_t & T) [inline], [static]
```

**12.288.2.55 zero()**

```
static INLINE CONST vect_t zero () [inline], [static], [inherited]
```

**12.288.3 Field Documentation****12.288.3.1 vect\_size**

```
const constexpr size_t vect_size = 16 [static], [constexpr]
```

**12.288.3.2 alignment**

```
const constexpr size_t alignment = 32 [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [simd256\\_int16.inl](#)

**12.289 Simd256\_impl< true, true, true, 4 > Struct Reference**

Inheritance diagram for Simd256\_impl< true, true, true, 4 >:



## Data Structures

- union [Converter](#)

## Public Types

- using vect\_t = \_\_m256i
- using half\_t = \_\_m128i
- using scalar\_t = int32\_t
- using simdHalf = Simd128< scalar\_t >
- using aligned\_allocator = AlignedAllocator< scalar\_t, Alignment(alignment)>
- using aligned\_vector = std::vector< scalar\_t, aligned\_allocator >
- template<class Field >
  - using is\_same\_element = std::is\_same< typename Field::Element, scalar\_t >
- using vect\_t = \_\_m512i
- using half\_t = \_\_m256i
- using scalar\_t = int32\_t
- using simdHalf = Simd256< scalar\_t >
- using aligned\_allocator = AlignedAllocator< scalar\_t, Alignment(alignment)>
- using aligned\_vector = std::vector< scalar\_t, aligned\_allocator >
- template<class Field >
  - using is\_same\_element = std::is\_same< typename Field::Element, scalar\_t >

## Static Public Member Functions

- static const std::string type\_string ()
- template<class T >
  - static constexpr bool valid (T \*p)
- template<class T >
  - static constexpr bool compliant (T n)
- static INLINE CONST vect\_t set1 (const scalar\_t x)
- static INLINE CONST vect\_t set (const scalar\_t x0, const scalar\_t x1, const scalar\_t x2, const scalar\_t x3, const scalar\_t x4, const scalar\_t x5, const scalar\_t x6, const scalar\_t x7)
- template<class T >
  - static INLINE PURE vect\_t gather (const scalar\_t \*const p, const T \*const idx)
- static INLINE PURE vect\_t load (const scalar\_t \*const p)
- static INLINE PURE vect\_t loadu (const scalar\_t \*const p)
- static INLINE void store (scalar\_t \*p, vect\_t v)
- static INLINE void storeu (scalar\_t \*p, vect\_t v)
- static INLINE void stream (scalar\_t \*p, const vect\_t v)
- template<int s>
  - static INLINE CONST vect\_t sll (const vect\_t a)
- template<int s>
  - static INLINE CONST vect\_t srl (const vect\_t a)
- template<int s>
  - static INLINE CONST vect\_t sra (const vect\_t a)
- template<uint8\_t s>
  - static INLINE CONST vect\_t shuffle\_twice (const vect\_t a)

- `template<uint32_t s>`  
`static INLINE CONST vect_t shuffle (const vect_t a)`
- `static INLINE CONST vect_t unpacklo_intrinsic (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t unpackhi_intrinsic (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t unpacklo (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t unpackhi (const vect_t a, const vect_t b)`
- `static INLINE void unpacklohi (vect_t &lo, vect_t &hi, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t pack_even (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t pack_odd (const vect_t a, const vect_t b)`
- `static INLINE void pack (vect_t &even, vect_t &odd, const vect_t a, const vect_t b)`
- `static INLINE void transpose (vect_t &r0, vect_t &r1, vect_t &r2, vect_t &r3, vect_t &r4, vect_t &r5, vect_t &r6, vect_t &r7)`
- `template<uint8_t s>`  
`static INLINE CONST vect_t blend (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t add (const vect_t a, const vect_t b)`
- `static INLINE vect_t addin (vect_t &a, const vect_t b)`
- `static INLINE CONST vect_t sub (const vect_t a, const vect_t b)`
- `static INLINE vect_t subin (vect_t &a, const vect_t b)`
- `static INLINE CONST vect_t mullo (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t mul (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t mulhi (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t mulx (vect_t a, vect_t b)`
- `static INLINE CONST vect_t fmadd (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmadin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmadx (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmadxin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fnmadd (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fnmadin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fnmaddx (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fnmaddxin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmsub (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmsubin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmsubx (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmsubxin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t eq (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t greater (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t lesser (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t greater_eq (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t lesser_eq (const vect_t a, const vect_t b)`
- `static INLINE CONST scalar_t hadd_to_scal (const vect_t a)`
- `static INLINE CONST vect_t round (const vect_t a)`
- `static INLINE vect_t mod (vect_t &C, const vect_t &P, const vect_t &INVP, const vect_t &NEGP, const vect_t &MIN, const vect_t &MAX, vect_t &Q, vect_t &T)`
- `static const std::string type_string ()`
- `template<class T>`  
`static constexpr bool valid (T *p)`
- `template<class T>`  
`static constexpr bool compliant (T n)`
- `static INLINE CONST vect_t set1 (const scalar_t x)`
- `static INLINE CONST vect_t set (const scalar_t x0, const scalar_t x1, const scalar_t x2, const scalar_t x3, const scalar_t x4, const scalar_t x5, const scalar_t x6, const scalar_t x7, const scalar_t x8, const scalar_t x9, const scalar_t x10, const scalar_t x11, const scalar_t x12, const scalar_t x13, const scalar_t x14, const scalar_t x15)`
- `template<class T>`  
`static INLINE PURE vect_t gather (const scalar_t *const p, const T *const idx)`

- static INLINE PURE vect\_t load (const scalar\_t \*const p)
- static INLINE PURE vect\_t loadu (const scalar\_t \*const p)
- static INLINE void store (scalar\_t \*p, vect\_t v)
- static INLINE void storeu (scalar\_t \*p, vect\_t v)
- static INLINE void stream (scalar\_t \*p, const vect\_t v)
- template<int s>  
static INLINE CONST vect\_t sll (const vect\_t a)
- template<int s>  
static INLINE CONST vect\_t srl (const vect\_t a)
- template<int s>  
static INLINE CONST vect\_t sra (const vect\_t a)
- template<uint8\_t s>  
static INLINE CONST vect\_t shuffle\_twice (const vect\_t a)
- template<uint64\_t s>  
static INLINE CONST vect\_t shuffle (const vect\_t a)
- static INLINE CONST vect\_t unpacklo\_intrinsic (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t unpackhi\_intrinsic (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t unpacklo (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t unpackhi (const vect\_t a, const vect\_t b)
- static INLINE void unpacklohi (vect\_t &lo, vect\_t &hi, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t pack\_even (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t pack\_odd (const vect\_t a, const vect\_t b)
- static INLINE void pack (vect\_t &even, vect\_t &odd, const vect\_t a, const vect\_t b)
- static INLINE void transpose (vect\_t &r0, vect\_t &r1, vect\_t &r2, vect\_t &r3, vect\_t &r4, vect\_t &r5, vect\_t &r6, vect\_t &r7, vect\_t &r8, vect\_t &r9, vect\_t &r10, vect\_t &r11, vect\_t &r12, vect\_t &r13, vect\_t &r14, vect\_t &r15)
- template<uint16\_t s>  
static INLINE CONST vect\_t blend (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t add (const vect\_t a, const vect\_t b)
- static INLINE vect\_t addin (vect\_t &a, const vect\_t b)
- static INLINE CONST vect\_t sub (const vect\_t a, const vect\_t b)
- static INLINE vect\_t subin (vect\_t &a, const vect\_t b)
- static INLINE CONST vect\_t mullo (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t mul (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t mulhi (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t mulx (vect\_t a, vect\_t b)
- static INLINE CONST vect\_t fmadd (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fmaddin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fmadddx (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fmadddxin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fnmadd (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fnmaddin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fnmaddx (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fnmaddxin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fmsub (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fmsubin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fmsubx (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fmsubxin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t eq (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t greater (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t lesser (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t greater\_eq (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t lesser\_eq (const vect\_t a, const vect\_t b)
- static INLINE CONST scalar\_t hadd\_to\_scal (const vect\_t a)
- static INLINE CONST vect\_t round (const vect\_t a)

- `static inline vect_t mod (vect_t &C, const vect_t &P, const vect_t &INVP, const vect_t &NEGP, const vect_t &MIN, const vect_t &MAX, vect_t &Q, vect_t &T)`
- `static inline const vect_t zero ()`
- `static inline const vect_t zero ()`
- `static inline const vect_t vor (const vect_t a, const vect_t b)`
- `static inline const vect_t vxor (const vect_t a, const vect_t b)`
- `static inline const vect_t vand (const vect_t a, const vect_t b)`
- `static inline const vect_t vandnot (const vect_t a, const vect_t b)`

#### Static Public Attributes

- `static const constexpr size_t vect_size = 8`
- `static const constexpr size_t alignment = 32`

### 12.289.1 Member Typedef Documentation

#### 12.289.1.1 vect\_t [1/2]

```
using vect_t = __m256i
```

#### 12.289.1.2 half\_t [1/2]

```
using half_t = __m128i
```

#### 12.289.1.3 scalar\_t [1/2]

```
using scalar_t = int32_t
```

#### 12.289.1.4 simdHalf [1/2]

```
using simdHalf = Simd128<scalar_t>
```

#### 12.289.1.5 aligned\_allocator [1/2]

```
using aligned_allocator = AlignedAllocator<scalar_t, Alignment(alignment)>
```

#### 12.289.1.6 aligned\_vector [1/2]

```
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```

#### 12.289.1.7 is\_same\_element [1/2]

```
template<class Field >
using is_same_element = std::is_same<typename Field::Element, scalar_t>
```

#### 12.289.1.8 vect\_t [2/2]

```
using vect_t = __m512i
```

#### 12.289.1.9 half\_t [2/2]

```
using half_t = __m256i
```

#### 12.289.1.10 scalar\_t [2/2]

```
using scalar_t = int32_t
```

#### 12.289.1.11 simdHalf [2/2]

```
using simdHalf = Simd256<scalar_t>
```

**12.289.1.12 aligned\_allocator [2/2]**

```
using aligned_allocator = AlignedAllocator<scalar_t, Alignment(alignment)>
```

**12.289.1.13 aligned\_vector [2/2]**

```
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```

**12.289.1.14 is\_same\_element [2/2]**

```
template<class Field >
using is_same_element = std::is_same<typename Field::Element, scalar_t>
```

**12.289.2 Member Function Documentation****12.289.2.1 type\_string() [1/2]**

```
static const std::string type_string () [inline], [static]
```

**12.289.2.2 valid() [1/2]**

```
template<class T >
static constexpr bool valid (
 T * p) [inline], [static], [constexpr]
```

**12.289.2.3 compliant() [1/2]**

```
template<class T >
static constexpr bool compliant (
 T n) [inline], [static], [constexpr]
```

**12.289.2.4 set1() [1/2]**

```
static INLINE CONST vect_t set1 (
 const scalar_t x) [inline], [static]
```

**12.289.2.5 set() [1/2]**

```
static INLINE CONST vect_t set (
 const scalar_t x0,
 const scalar_t x1,
 const scalar_t x2,
 const scalar_t x3,
 const scalar_t x4,
 const scalar_t x5,
 const scalar_t x6,
 const scalar_t x7) [inline], [static]
```

**12.289.2.6 gather() [1/2]**

```
template<class T >
static INLINE PURE vect_t gather (
 const scalar_t *const p,
 const T *const idx) [inline], [static]
```

**12.289.2.7 load() [1/2]**

```
static INLINE PURE vect_t load (
 const scalar_t *const p) [inline], [static]
```

**12.289.2.8 loadu()** [1/2]

```
static INLINE PURE vect_t loadu (
 const scalar_t *const p) [inline], [static]
```

**12.289.2.9 store()** [1/2]

```
static INLINE void store (
 scalar_t * p,
 vect_t v) [inline], [static]
```

**12.289.2.10 storeu()** [1/2]

```
static INLINE void storeu (
 scalar_t * p,
 vect_t v) [inline], [static]
```

**12.289.2.11 stream()** [1/2]

```
static INLINE void stream (
 scalar_t * p,
 const vect_t v) [inline], [static]
```

**12.289.2.12 sll()** [1/2]

```
template<int s>
static INLINE CONST vect_t sll (
 const vect_t a) [inline], [static]
```

**12.289.2.13 srl()** [1/2]

```
template<int s>
static INLINE CONST vect_t srl (
 const vect_t a) [inline], [static]
```

**12.289.2.14 sra()** [1/2]

```
template<int s>
static INLINE CONST vect_t sra (
 const vect_t a) [inline], [static]
```

**12.289.2.15 shuffle\_twice()** [1/2]

```
template<uint8_t s>
static INLINE CONST vect_t shuffle_twice (
 const vect_t a) [inline], [static]
```

**12.289.2.16 shuffle()** [1/2]

```
template<uint32_t s>
static INLINE CONST vect_t shuffle (
 const vect_t a) [inline], [static]
```

**12.289.2.17 unpacklo\_intrinsic()** [1/2]

```
static INLINE CONST vect_t unpacklo_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static]
```



**12.289.2.18 unpackhi\_intrinsic()** [1/2]

```
static INLINE CONST vect_t unpackhi_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.19 unpacklo()** [1/2]

```
static INLINE CONST vect_t unpacklo (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.20 unpackhi()** [1/2]

```
static INLINE CONST vect_t unpackhi (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.21 unpacklohi()** [1/2]

```
static INLINE void unpacklohi (
 vect_t & lo,
 vect_t & hi,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.22 pack\_even()** [1/2]

```
static INLINE CONST vect_t pack_even (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.23 pack\_odd()** [1/2]

```
static INLINE CONST vect_t pack_odd (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.24 pack()** [1/2]

```
static INLINE void pack (
 vect_t & even,
 vect_t & odd,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.25 transpose()** [1/2]

```
static INLINE void transpose (
 vect_t & r0,
 vect_t & r1,
 vect_t & r2,
 vect_t & r3,
 vect_t & r4,
 vect_t & r5,
 vect_t & r6,
 vect_t & r7) [inline], [static]
```

**12.289.2.26 blend()** [1/2]

```
template<uint8_t s>
static INLINE CONST vect_t blend (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.27 add()** [1/2]

```
static INLINE CONST vect_t add (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.28 addin()** [1/2]

```
static INLINE vect_t addin (
 vect_t & a,
 const vect_t b) [inline], [static]
```

**12.289.2.29 sub()** [1/2]

```
static INLINE CONST vect_t sub (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.30 subin()** [1/2]

```
static INLINE vect_t subin (
 vect_t & a,
 const vect_t b) [inline], [static]
```

**12.289.2.31 mullo()** [1/2]

```
static INLINE CONST vect_t mullo (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.32 mul()** [1/2]

```
static INLINE CONST vect_t mul (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.33 mulhi()** [1/2]

```
static INLINE CONST vect_t mulhi (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.34 mulx()** [1/2]

```
static INLINE CONST vect_t mulx (
 vect_t a,
 vect_t b) [inline], [static]
```

**12.289.2.35 fmadd()** [1/2]

```
static INLINE CONST vect_t fmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.36 fmaddin()** [1/2]

```
static INLINE vect_t fmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.37 fmaddx()** [1/2]

```
static INLINE CONST vect_t fmaddx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.38 fmaddxin()** [1/2]

```
static INLINE vect_t fmaddxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.39 fnmadd()** [1/2]

```
static INLINE CONST vect_t fnmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.40 fnmaddin()** [1/2]

```
static INLINE vect_t fnmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.41 fnmaddx()** [1/2]

```
static INLINE CONST vect_t fnmaddx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.42 fnmaddxin()** [1/2]

```
static INLINE vect_t fnmaddxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.43 fmsub()** [1/2]

```
static INLINE CONST vect_t fmsub (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.44 fmsubin()** [1/2]

```
static INLINE vect_t fmsubin (
 vect_t & c,
```

```
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.289.2.45 fmsubx() [1/2]

```
static INLINE CONST vect_t fmsubx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.289.2.46 fmsubxin() [1/2]

```
static INLINE vect_t fmsubxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.289.2.47 eq() [1/2]

```
static INLINE CONST vect_t eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.289.2.48 greater() [1/2]

```
static INLINE CONST vect_t greater (
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.289.2.49 lesser() [1/2]

```
static INLINE CONST vect_t lesser (
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.289.2.50 greater\_eq() [1/2]

```
static INLINE CONST vect_t greater_eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.289.2.51 lesser\_eq() [1/2]

```
static INLINE CONST vect_t lesser_eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.289.2.52 hadd\_to\_scal() [1/2]

```
static INLINE CONST scalar_t hadd_to_scal (
 const vect_t a) [inline], [static]
```

#### 12.289.2.53 round() [1/2]

```
static INLINE CONST vect_t round (
 const vect_t a) [inline], [static]
```

**12.289.2.54 mod()** [1/2]

```
static INLINE vect_t mod (
 vect_t & C,
 const vect_t & P,
 const vect_t & INVP,
 const vect_t & NEGP,
 const vect_t & MIN,
 const vect_t & MAX,
 vect_t & Q,
 vect_t & T) [inline], [static]
```

**12.289.2.55 type\_string()** [2/2]

```
static const std::string type_string () [inline], [static]
```

**12.289.2.56 valid()** [2/2]

```
template<class T >
static constexpr bool valid (
 T * p) [inline], [static], [constexpr]
```

**12.289.2.57 compliant()** [2/2]

```
template<class T >
static constexpr bool compliant (
 T n) [inline], [static], [constexpr]
```

**12.289.2.58 set1()** [2/2]

```
static INLINE CONST vect_t set1 (
 const scalar_t x) [inline], [static]
```

**12.289.2.59 set()** [2/2]

```
static INLINE CONST vect_t set (
 const scalar_t x0,
 const scalar_t x1,
 const scalar_t x2,
 const scalar_t x3,
 const scalar_t x4,
 const scalar_t x5,
 const scalar_t x6,
 const scalar_t x7,
 const scalar_t x8,
 const scalar_t x9,
 const scalar_t x10,
 const scalar_t x11,
 const scalar_t x12,
 const scalar_t x13,
 const scalar_t x14,
 const scalar_t x15) [inline], [static]
```

**12.289.2.60 gather()** [2/2]

```
template<class T >
static INLINE PURE vect_t gather (
 const scalar_t *const p,
 const T *const idx) [inline], [static]
```

**12.289.2.61 load()** [2/2]

```
static INLINE PURE vect_t load (
 const scalar_t *const p) [inline], [static]
```

**12.289.2.62 loadu()** [2/2]

```
static INLINE PURE vect_t loadu (
 const scalar_t *const p) [inline], [static]
```

**12.289.2.63 store()** [2/2]

```
static INLINE void store (
 scalar_t * p,
 vect_t v) [inline], [static]
```

**12.289.2.64 storeu()** [2/2]

```
static INLINE void storeu (
 scalar_t * p,
 vect_t v) [inline], [static]
```

**12.289.2.65 stream()** [2/2]

```
static INLINE void stream (
 scalar_t * p,
 const vect_t v) [inline], [static]
```

**12.289.2.66 sll()** [2/2]

```
template<int s>
static INLINE CONST vect_t sll (
 const vect_t a) [inline], [static]
```

**12.289.2.67 srl()** [2/2]

```
template<int s>
static INLINE CONST vect_t srl (
 const vect_t a) [inline], [static]
```

**12.289.2.68 sra()** [2/2]

```
template<int s>
static INLINE CONST vect_t sra (
 const vect_t a) [inline], [static]
```

**12.289.2.69 shuffle\_twice()** [2/2]

```
template<uint8_t s>
static INLINE CONST vect_t shuffle_twice (
 const vect_t a) [inline], [static]
```

**12.289.2.70 shuffle()** [2/2]

```
template<uint64_t s>
static INLINE CONST vect_t shuffle (
 const vect_t a) [inline], [static]
```

**12.289.2.71 unpacklo\_intrinsic()** [2/2]

```
static INLINE CONST vect_t unpacklo_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.72 unpackhi\_intrinsic()** [2/2]

```
static INLINE CONST vect_t unpackhi_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.73 unpacklo()** [2/2]

```
static INLINE CONST vect_t unpacklo (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.74 unpackhi()** [2/2]

```
static INLINE CONST vect_t unpackhi (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.75 unpacklohi()** [2/2]

```
static INLINE void unpacklohi (
 vect_t & lo,
 vect_t & hi,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.76 pack\_even()** [2/2]

```
static INLINE CONST vect_t pack_even (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.77 pack\_odd()** [2/2]

```
static INLINE CONST vect_t pack_odd (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.78 pack()** [2/2]

```
static INLINE void pack (
 vect_t & even,
 vect_t & odd,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.79 transpose()** [2/2]

```
static INLINE void transpose (
 vect_t & r0,
 vect_t & r1,
 vect_t & r2,
 vect_t & r3,
 vect_t & r4,
```

```

 vect_t & r5,
 vect_t & r6,
 vect_t & r7,
 vect_t & r8,
 vect_t & r9,
 vect_t & r10,
 vect_t & r11,
 vect_t & r12,
 vect_t & r13,
 vect_t & r14,
 vect_t & r15) [inline], [static]

```

#### 12.289.2.80 blend() [2/2]

```

template<uint16_t s>
static INLINE CONST vect_t blend (
 const vect_t a,
 const vect_t b) [inline], [static]

```

#### 12.289.2.81 add() [2/2]

```

static INLINE CONST vect_t add (
 const vect_t a,
 const vect_t b) [inline], [static]

```

#### 12.289.2.82 addin() [2/2]

```

static INLINE vect_t addin (
 vect_t & a,
 const vect_t b) [inline], [static]

```

#### 12.289.2.83 sub() [2/2]

```

static INLINE CONST vect_t sub (
 const vect_t a,
 const vect_t b) [inline], [static]

```

#### 12.289.2.84 subin() [2/2]

```

static INLINE vect_t subin (
 vect_t & a,
 const vect_t b) [inline], [static]

```

#### 12.289.2.85 mullo() [2/2]

```

static INLINE CONST vect_t mullo (
 const vect_t a,
 const vect_t b) [inline], [static]

```

#### 12.289.2.86 mul() [2/2]

```

static INLINE CONST vect_t mul (
 const vect_t a,
 const vect_t b) [inline], [static]

```

#### 12.289.2.87 mulhi() [2/2]

```

static INLINE CONST vect_t mulhi (
 const vect_t a,
 const vect_t b) [inline], [static]

```



**12.289.2.88 mulx()** [2/2]

```
static INLINE CONST vect_t mulx (
 vect_t a,
 vect_t b) [inline], [static]
```

**12.289.2.89 fmadd()** [2/2]

```
static INLINE CONST vect_t fmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.90 fmaddin()** [2/2]

```
static INLINE vect_t fmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.91 fmaddx()** [2/2]

```
static INLINE CONST vect_t fmaddx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.92 fmaddxin()** [2/2]

```
static INLINE vect_t fmaddxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.93 fnmadd()** [2/2]

```
static INLINE CONST vect_t fnmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.94 fnmaddin()** [2/2]

```
static INLINE vect_t fnmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.95 fnmaddx()** [2/2]

```
static INLINE CONST vect_t fnmaddx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.96 fnmaddxin()** [2/2]

```
static INLINE vect_t fnmaddxin (
 vect_t & c,
```

```
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.289.2.97 fmsub() [2/2]

```
static INLINE CONST vect_t fmsub (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.289.2.98 fmsubin() [2/2]

```
static INLINE vect_t fmsubin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.289.2.99 fmsubx() [2/2]

```
static INLINE CONST vect_t fmsubx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.289.2.100 fmsubxin() [2/2]

```
static INLINE vect_t fmsubxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.289.2.101 eq() [2/2]

```
static INLINE CONST vect_t eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.289.2.102 greater() [2/2]

```
static INLINE CONST vect_t greater (
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.289.2.103 lesser() [2/2]

```
static INLINE CONST vect_t lesser (
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.289.2.104 greater\_eq() [2/2]

```
static INLINE CONST vect_t greater_eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.289.2.105 lesser\_eq() [2/2]

```
static INLINE CONST vect_t lesser_eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.289.2.106 hadd\_to\_scal()** [2/2]

```
static INLINE CONST scalar_t hadd_to_scal (
 const vect_t a) [inline], [static]
```

**12.289.2.107 round()** [2/2]

```
static INLINE CONST vect_t round (
 const vect_t a) [inline], [static]
```

**12.289.2.108 mod()** [2/2]

```
static INLINE vect_t mod (
 vect_t & C,
 const vect_t & P,
 const vect_t & INVP,
 const vect_t & NEGP,
 const vect_t & MIN,
 const vect_t & MAX,
 vect_t & Q,
 vect_t & T) [inline], [static]
```

**12.289.2.109 zero()** [1/2]

```
static INLINE CONST vect_t zero () [inline], [static], [inherited]
```

**12.289.2.110 zero()** [2/2]

```
static INLINE CONST vect_t zero () [inline], [static], [inherited]
```

**12.289.2.111 vor()**

```
static INLINE CONST vect_t vor (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.289.2.112 vxor()**

```
static INLINE CONST vect_t vxor (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.289.2.113 vand()**

```
static INLINE CONST vect_t vand (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.289.2.114 vandnot()**

```
static INLINE CONST vect_t vandnot (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.289.3 Field Documentation****12.289.3.1 vect\_size**

```
static const constexpr size_t vect_size = 8 [static], [constexpr]
```

### 12.289.3.2 alignment

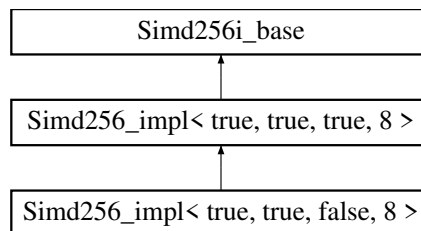
```
static const constexpr size_t alignment = 32 [static], [constexpr]
```

The documentation for this struct was generated from the following files:

- [simd256\\_int32.inl](#)
- [simd512\\_int32.inl](#)

## 12.290 Simd256\_impl< true, true, true, 8 > Struct Reference

Inheritance diagram for Simd256\_impl< true, true, true, 8 >:



### Data Structures

- union [Converter](#)

### Public Types

- using vect\_t = \_\_m256i
- using half\_t = \_\_m128i
- using scalar\_t = int64\_t
- using simdHalf = Simd128< scalar\_t >
- using aligned\_allocator = AlignedAllocator< scalar\_t, Alignment(alignment)>
- using aligned\_vector = std::vector< scalar\_t, aligned\_allocator >
- template<class Field >  
using is\_same\_element = std::is\_same< typename Field::Element, scalar\_t >

### Static Public Member Functions

- static const std::string type\_string ()
- template<class T >  
static constexpr bool valid (T \*p)
- template<class T >  
static constexpr bool compliant (T n)
- static INLINE CONST vect\_t set1 (const scalar\_t x)
- static INLINE CONST vect\_t set (const scalar\_t x0, const scalar\_t x1, const scalar\_t x2, const scalar\_t x3)
- template<class T >  
static INLINE PURE vect\_t gather (const scalar\_t \*const p, const T \*const idx)
- template<int idx>  
static INLINE CONST scalar\_t get (vect\_t v)
- static INLINE PURE vect\_t load (const scalar\_t \*const p)
- static INLINE PURE vect\_t loadu (const scalar\_t \*const p)
- static INLINE void store (scalar\_t \*p, vect\_t v)
- static INLINE void storeu (scalar\_t \*p, vect\_t v)
- static INLINE void stream (scalar\_t \*p, const vect\_t v)
- template<int s>  
static INLINE CONST vect\_t sll (const vect\_t a)

- `template<int s>`  
`static INLINE CONST vect_t srl (const vect_t a)`
- `template<int s>`  
`static INLINE CONST vect_t sra (const vect_t a)`
- `template<uint8_t s>`  
`static INLINE CONST vect_t shuffle (const vect_t a)`
- `static INLINE CONST vect_t unpacklo_intrinsic (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t unpackhi_intrinsic (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t unpacklo (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t unpackhi (const vect_t a, const vect_t b)`
- `static INLINE void unpacklohi (vect_t &lo, vect_t &hi, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t pack_even (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t pack_odd (const vect_t a, const vect_t b)`
- `static INLINE void pack (vect_t &even, vect_t &odd, const vect_t a, const vect_t b)`
- `static INLINE void transpose (vect_t &r0, vect_t &r1, vect_t &r2, vect_t &r3)`
- `template<uint8_t s>`  
`static INLINE CONST vect_t blend (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t add (const vect_t a, const vect_t b)`
- `static INLINE vect_t addin (vect_t &a, const vect_t b)`
- `static INLINE CONST vect_t sub (const vect_t a, const vect_t b)`
- `static INLINE vect_t subin (vect_t &a, const vect_t b)`
- `static INLINE CONST vect_t mullo (vect_t a, vect_t b)`
- `static INLINE CONST vect_t mul (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t mulhi (vect_t a, vect_t b)`
- `static INLINE CONST vect_t mulx (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmadd (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmaddin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmadx (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmadxin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fnmadd (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fnmaddin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fnmadx (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fnmadxin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmsub (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmsubin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmsubx (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmsubxin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t eq (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t greater (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t lesser (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t greater_eq (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t lesser_eq (const vect_t a, const vect_t b)`
- `static INLINE CONST scalar_t hadd_to_scal (const vect_t a)`
- `static INLINE CONST vect_t round (const vect_t a)`
- `static INLINE CONST vect_t mask_high ()`
- `static INLINE CONST vect_t mulhi_fast (vect_t x, vect_t y)`
- `static INLINE vect_t mod (vect_t &C, const __m256d &P, const __m256d &INVP, const __m256d &NEGP, const vect_t &POW50REM, const __m256d &MIN, const __m256d &MAX, __m256d &Q, __m256d &T)`
- `static INLINE CONST vect_t zero ()`

### Static Public Attributes

- `static const constexpr size_t vect_size = 4`
- `static const constexpr size_t alignment = 32`

## Static Protected Member Functions

- `static INLINE CONST vect_t signbits (const vect_t x)`

## 12.290.1 Member Typedef Documentation

### 12.290.1.1 vect\_t

```
using vect_t = __m256i
```

### 12.290.1.2 half\_t

```
using half_t = __m128i
```

### 12.290.1.3 scalar\_t

```
using scalar_t = int64_t
```

### 12.290.1.4 simdHalf

```
using simdHalf = Simd128<scalar_t>
```

### 12.290.1.5 aligned\_allocator

```
using aligned_allocator = AlignedAllocator<scalar_t, Alignment (alignment)>
```

### 12.290.1.6 aligned\_vector

```
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```

### 12.290.1.7 is\_same\_element

```
template<class Field >
using is_same_element = std::is_same<typename Field::Element, scalar_t>
```

## 12.290.2 Member Function Documentation

### 12.290.2.1 type\_string()

```
static const std::string type_string () [inline], [static]
```

### 12.290.2.2 valid()

```
template<class T >
static constexpr bool valid (
 T * p) [inline], [static], [constexpr]
```

### 12.290.2.3 compliant()

```
template<class T >
static constexpr bool compliant (
 T n) [inline], [static], [constexpr]
```

### 12.290.2.4 set1()

```
static INLINE CONST vect_t set1 (
 const scalar_t x) [inline], [static]
```

**12.290.2.5 set()**

```
static INLINE CONST vect_t set (
 const scalar_t x0,
 const scalar_t x1,
 const scalar_t x2,
 const scalar_t x3) [inline], [static]
```

**12.290.2.6 gather()**

```
template<class T >
static INLINE PURE vect_t gather (
 const scalar_t *const p,
 const T *const idx) [inline], [static]
```

**12.290.2.7 get()**

```
template<int idx>
static INLINE CONST scalar_t get (
 vect_t v) [inline], [static]
```

**12.290.2.8 load()**

```
static INLINE PURE vect_t load (
 const scalar_t *const p) [inline], [static]
```

**12.290.2.9 loadu()**

```
static INLINE PURE vect_t loadu (
 const scalar_t *const p) [inline], [static]
```

**12.290.2.10 store()**

```
static INLINE void store (
 scalar_t * p,
 vect_t v) [inline], [static]
```

**12.290.2.11 storeu()**

```
static INLINE void storeu (
 scalar_t * p,
 vect_t v) [inline], [static]
```

**12.290.2.12 stream()**

```
static INLINE void stream (
 scalar_t * p,
 const vect_t v) [inline], [static]
```

**12.290.2.13 sll()**

```
template<int s>
static INLINE CONST vect_t sll (
 const vect_t a) [inline], [static]
```

**12.290.2.14 srl()**

```
template<int s>
static INLINE CONST vect_t srl (
 const vect_t a) [inline], [static]
```

**12.290.2.15 sra()**

```
template<int s>
static INLINE CONST vect_t sra (
 const vect_t a) [inline], [static]
```

**12.290.2.16 shuffle()**

```
template<uint8_t s>
static INLINE CONST vect_t shuffle (
 const vect_t a) [inline], [static]
```

**12.290.2.17 unpacklo\_intrinsic()**

```
static INLINE CONST vect_t unpacklo_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.290.2.18 unpackhi\_intrinsic()**

```
static INLINE CONST vect_t unpackhi_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.290.2.19 unpacklo()**

```
static INLINE CONST vect_t unpacklo (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.290.2.20 unpackhi()**

```
static INLINE CONST vect_t unpackhi (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.290.2.21 unpacklohi()**

```
static INLINE void unpacklohi (
 vect_t & lo,
 vect_t & hi,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.290.2.22 pack\_even()**

```
static INLINE CONST vect_t pack_even (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.290.2.23 pack\_odd()**

```
static INLINE CONST vect_t pack_odd (
 const vect_t a,
 const vect_t b) [inline], [static]
```



**12.290.2.24 pack()**

```
static INLINE void pack (
 vect_t & even,
 vect_t & odd,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.290.2.25 transpose()**

```
static INLINE void transpose (
 vect_t & r0,
 vect_t & r1,
 vect_t & r2,
 vect_t & r3) [inline], [static]
```

**12.290.2.26 blend()**

```
template<uint8_t s>
static INLINE CONST vect_t blend (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.290.2.27 add()**

```
static INLINE CONST vect_t add (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.290.2.28 addin()**

```
static INLINE vect_t addin (
 vect_t & a,
 const vect_t b) [inline], [static]
```

**12.290.2.29 sub()**

```
static INLINE CONST vect_t sub (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.290.2.30 subin()**

```
static INLINE vect_t subin (
 vect_t & a,
 const vect_t b) [inline], [static]
```

**12.290.2.31 mullo()**

```
static INLINE CONST vect_t mullo (
 vect_t a,
 vect_t b) [inline], [static]
```

**12.290.2.32 mul()**

```
static INLINE CONST vect_t mul (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.290.2.33 mulhi()**

```
static INLINE CONST vect_t mulhi (
 vect_t a,
 vect_t b) [inline], [static]
```

**12.290.2.34 mulx()**

```
static INLINE CONST vect_t mulx (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.290.2.35 fmadd()**

```
static INLINE CONST vect_t fmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.290.2.36 fmaddin()**

```
static INLINE vect_t fmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.290.2.37 fmaddx()**

```
static INLINE CONST vect_t fmaddx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.290.2.38 fmaddxin()**

```
static INLINE vect_t fmaddxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.290.2.39 fnmadd()**

```
static INLINE CONST vect_t fnmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.290.2.40 fnmaddin()**

```
static INLINE vect_t fnmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.290.2.41 fnmaddx()**

```
static INLINE CONST vect_t fnmaddx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.290.2.42 fnmaddxin()**

```
static INLINE vect_t fnmaddxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.290.2.43 fmsub()**

```
static INLINE CONST vect_t fmsub (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.290.2.44 fmsubin()**

```
static INLINE vect_t fmsubin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.290.2.45 fmsubx()**

```
static INLINE CONST vect_t fmsubx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.290.2.46 fmsubxin()**

```
static INLINE vect_t fmsubxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.290.2.47 eq()**

```
static INLINE CONST vect_t eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.290.2.48 greater()**

```
static INLINE CONST vect_t greater (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.290.2.49 lesser()**

```
static INLINE CONST vect_t lesser (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.290.2.50 greater\_eq()**

```
static INLINE CONST vect_t greater_eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.290.2.51 lesser\_eq()**

```
static INLINE CONST vect_t lesser_eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.290.2.52 hadd\_to\_scal()**

```
static INLINE CONST scalar_t hadd_to_scal (
 const vect_t a) [inline], [static]
```

**12.290.2.53 round()**

```
static INLINE CONST vect_t round (
 const vect_t a) [inline], [static]
```

**12.290.2.54 mask\_high()**

```
static INLINE CONST vect_t mask_high () [inline], [static]
```

**12.290.2.55 mulhi\_fast()**

```
INLINE CONST vect_t mulhi_fast (
 vect_t x,
 vect_t y) [static]
```

**12.290.2.56 mod()**

```
INLINE vect_t mod (
 vect_t & C,
 const __m256d & P,
 const __m256d & INVP,
 const __m256d & NEGP,
 const vect_t & POW5OREM,
 const __m256d & MIN,
 const __m256d & MAX,
 __m256d & Q,
 __m256d & T) [static]
```

**12.290.2.57 signbits()**

```
static INLINE CONST vect_t signbits (
 const vect_t x) [inline], [static], [protected]
```

**12.290.2.58 zero()**

```
static INLINE CONST vect_t zero () [inline], [static], [inherited]
```

**12.290.3 Field Documentation****12.290.3.1 vect\_size**

```
const constexpr size_t vect_size = 4 [static], [constexpr]
```

**12.290.3.2 alignment**

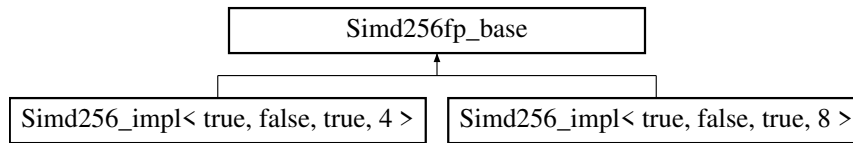
```
const constexpr size_t alignment = 32 [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [simd256\\_int64.inl](#)

## 12.291 Simd256fp\_base Struct Reference

Inheritance diagram for Simd256fp\_base:



The documentation for this struct was generated from the following file:

- [simd256.inl](#)

## 12.292 Simd256i\_base Struct Reference

Inheritance diagram for Simd256i\_base:



### Public Types

- `using vect_t = __m256i`

### Static Public Member Functions

- `static INLINE CONST vect_t zero ()`

### 12.292.1 Member Typedef Documentation

#### 12.292.1.1 vect\_t

```
using vect_t = __m256i
```

### 12.292.2 Member Function Documentation

#### 12.292.2.1 zero()

```
static INLINE CONST vect_t zero () [inline], [static]
```

The documentation for this struct was generated from the following file:

- [simd256.inl](#)

## 12.293 Simd512\_impl< ArithType, Int, Signed, Size > Struct Template Reference

The documentation for this struct was generated from the following file:

- [simd512.inl](#)

## 12.294 Simd512\_impl< true, false, true, 4 > Struct Reference

The documentation for this struct was generated from the following file:

- [simd512\\_float.inl](#)

## 12.295 Simd512\_impl< true, false, true, 8 > Struct Reference

### Public Types

- using vect\_t = \_\_m512d
- using scalar\_t = double
- using aligned\_allocator = AlignedAllocator< scalar\_t, Alignment(alignment)>
- using aligned\_vector = std::vector< scalar\_t, aligned\_allocator >
- template<class Field >  
using is\_same\_element = std::is\_same< typename Field::Element, scalar\_t >

### Static Public Member Functions

- static const std::string type\_string ()
- template<class T >  
static constexpr bool valid (T \*p)
- template<class T >  
static constexpr bool compliant (T n)
- static INLINE CONST vect\_t zero ()
- static INLINE CONST vect\_t set1 (const scalar\_t x)
- static INLINE CONST vect\_t set (const scalar\_t x1, const scalar\_t x2, const scalar\_t x3, const scalar\_t x4, const scalar\_t x5, const scalar\_t x6, const scalar\_t x7, const scalar\_t x8)
- template<class T >  
static INLINE PURE vect\_t gather (const scalar\_t \*const p, const T \*const idx)
- static INLINE PURE vect\_t load (const scalar\_t \*const p)
- static INLINE PURE vect\_t loadu (const scalar\_t \*const p)
- static INLINE void store (const scalar\_t \*p, const vect\_t v)
- static INLINE void storeu (const scalar\_t \*p, const vect\_t v)
- static INLINE void stream (const scalar\_t \*p, const vect\_t v)
- template<uint8\_t s>  
static INLINE CONST vect\_t shuffle (const vect\_t a)
- static INLINE CONST vect\_t unpacklo\_intrinsic (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t unpackhi\_intrinsic (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t unpacklo (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t unpackhi (const vect\_t a, const vect\_t b)
- static INLINE void unpacklohi (vect\_t &lo, vect\_t &hi, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t pack\_even (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t pack\_odd (const vect\_t a, const vect\_t b)
- static INLINE void pack (vect\_t &even, vect\_t &odd, const vect\_t a, const vect\_t b)
- static INLINE void transpose (vect\_t &r0, vect\_t &r1, vect\_t &r2, vect\_t &r3, vect\_t &r4, vect\_t &r5, vect\_t &r6, vect\_t &r7)
- template<uint8\_t s>  
static INLINE CONST vect\_t blend (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t blendv (const vect\_t a, const vect\_t b, const vect\_t mask)
- static INLINE CONST vect\_t add (const vect\_t a, const vect\_t b)
- static INLINE vect\_t addin (vect\_t &a, const vect\_t b)
- static INLINE CONST vect\_t sub (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t subin (vect\_t &a, const vect\_t b)
- static INLINE CONST vect\_t mul (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t mulin (vect\_t &a, const vect\_t b)
- static INLINE CONST vect\_t div (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fmadd (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fmaddin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fnmadd (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fnmaddin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fmsub (const vect\_t c, const vect\_t a, const vect\_t b)

- `static inline const vect_t fmsubin (vect_t &c, const vect_t a, const vect_t b)`
- `static inline const vect_t eq (const vect_t a, const vect_t b)`
- `static inline const vect_t lesser (const vect_t a, const vect_t b)`
- `static inline const vect_t lesser_eq (const vect_t a, const vect_t b)`
- `static inline const vect_t greater (const vect_t a, const vect_t b)`
- `static inline const vect_t greater_eq (const vect_t a, const vect_t b)`
- `static inline const vect_t floor (const vect_t a)`
- `static inline const vect_t ceil (const vect_t a)`
- `static inline const vect_t round (const vect_t a)`
- `static inline const vect_t hadd (const vect_t a, const vect_t b)`
- `static inline const scalar_t hadd_to_scal (const vect_t a)`

### Static Public Attributes

- `static const constexpr size_t vect_size = 8`
- `static const constexpr size_t alignment = 64`

## 12.295.1 Member Typedef Documentation

### 12.295.1.1 vect\_t

```
using vect_t = __m512d
```

### 12.295.1.2 scalar\_t

```
using scalar_t = double
```

### 12.295.1.3 aligned\_allocator

```
using aligned_allocator = AlignedAllocator<scalar_t, Alignment(alignment)>
```

### 12.295.1.4 aligned\_vector

```
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```

### 12.295.1.5 is\_same\_element

```
template<class Field >
using is_same_element = std::is_same<typename Field::Element, scalar_t>
```

## 12.295.2 Member Function Documentation

### 12.295.2.1 type\_string()

```
static const std::string type_string () [inline], [static]
```

### 12.295.2.2 valid()

```
template<class T >
static constexpr bool valid (
 T * p) [inline], [static], [constexpr]
```

### 12.295.2.3 compliant()

```
template<class T >
static constexpr bool compliant (
 T n) [inline], [static], [constexpr]
```

**12.295.2.4 zero()**

```
static INLINE CONST vect_t zero () [inline], [static]
```

**12.295.2.5 set1()**

```
static INLINE CONST vect_t set1 (
 const scalar_t x) [inline], [static]
```

**12.295.2.6 set()**

```
static INLINE CONST vect_t set (
 const scalar_t x1,
 const scalar_t x2,
 const scalar_t x3,
 const scalar_t x4,
 const scalar_t x5,
 const scalar_t x6,
 const scalar_t x7,
 const scalar_t x8) [inline], [static]
```

**12.295.2.7 gather()**

```
template<class T >
static INLINE PURE vect_t gather (
 const scalar_t *const p,
 const T *const idx) [inline], [static]
```

**12.295.2.8 load()**

```
static INLINE PURE vect_t load (
 const scalar_t *const p) [inline], [static]
```

**12.295.2.9 loadu()**

```
static INLINE PURE vect_t loadu (
 const scalar_t *const p) [inline], [static]
```

**12.295.2.10 store()**

```
static INLINE void store (
 const scalar_t * p,
 const vect_t v) [inline], [static]
```

**12.295.2.11 storeu()**

```
static INLINE void storeu (
 const scalar_t * p,
 const vect_t v) [inline], [static]
```

**12.295.2.12 stream()**

```
static INLINE void stream (
 const scalar_t * p,
 const vect_t v) [inline], [static]
```

**12.295.2.13 shuffle()**

```
template<uint8_t s>
static INLINE CONST vect_t shuffle (
 const vect_t a) [inline], [static]
```



**12.295.2.14 unpacklo\_intrinsic()**

```
static INLINE CONST vect_t unpacklo_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.295.2.15 unpackhi\_intrinsic()**

```
static INLINE CONST vect_t unpackhi_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.295.2.16 unpacklo()**

```
static INLINE CONST vect_t unpacklo (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.295.2.17 unpackhi()**

```
static INLINE CONST vect_t unpackhi (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.295.2.18 unpacklohi()**

```
static INLINE void unpacklohi (
 vect_t & lo,
 vect_t & hi,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.295.2.19 pack\_even()**

```
static INLINE CONST vect_t pack_even (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.295.2.20 pack\_odd()**

```
static INLINE CONST vect_t pack_odd (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.295.2.21 pack()**

```
static INLINE void pack (
 vect_t & even,
 vect_t & odd,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.295.2.22 transpose()**

```
static INLINE void transpose (
 vect_t & r0,
 vect_t & r1,
 vect_t & r2,
 vect_t & r3,
 vect_t & r4,
```

```
 vect_t & r5,
 vect_t & r6,
 vect_t & r7) [inline], [static]
```

#### 12.295.2.23 blend()

```
template<uint8_t s>
static INLINE CONST vect_t blend (
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.295.2.24 blendv()

```
static INLINE CONST vect_t blendv (
 const vect_t a,
 const vect_t b,
 const vect_t mask) [inline], [static]
```

#### 12.295.2.25 add()

```
static INLINE CONST vect_t add (
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.295.2.26 addin()

```
static INLINE vect_t addin (
 vect_t & a,
 const vect_t b) [inline], [static]
```

#### 12.295.2.27 sub()

```
static INLINE CONST vect_t sub (
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.295.2.28 subin()

```
static INLINE CONST vect_t subin (
 vect_t & a,
 const vect_t b) [inline], [static]
```

#### 12.295.2.29 mul()

```
static INLINE CONST vect_t mul (
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.295.2.30 mulin()

```
static INLINE CONST vect_t mulin (
 vect_t & a,
 const vect_t b) [inline], [static]
```

#### 12.295.2.31 div()

```
static INLINE CONST vect_t div (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.295.2.32 fmadd()**

```
static INLINE CONST vect_t fmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.295.2.33 fmaddin()**

```
static INLINE CONST vect_t fmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.295.2.34 fnmadd()**

```
static INLINE CONST vect_t fnmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.295.2.35 fnmaddin()**

```
static INLINE CONST vect_t fnmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.295.2.36 fmsub()**

```
static INLINE CONST vect_t fmsub (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.295.2.37 fmsubin()**

```
static INLINE CONST vect_t fmsubin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.295.2.38 eq()**

```
static INLINE CONST vect_t eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.295.2.39 lesser()**

```
static INLINE CONST vect_t lesser (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.295.2.40 lesser\_eq()**

```
static INLINE CONST vect_t lesser_eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.295.2.41 greater()**

```
static INLINE CONST vect_t greater (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.295.2.42 greater\_eq()**

```
static INLINE CONST vect_t greater_eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.295.2.43 floor()**

```
static INLINE CONST vect_t floor (
 const vect_t a) [inline], [static]
```

**12.295.2.44 ceil()**

```
static INLINE CONST vect_t ceil (
 const vect_t a) [inline], [static]
```

**12.295.2.45 round()**

```
static INLINE CONST vect_t round (
 const vect_t a) [inline], [static]
```

**12.295.2.46 hadd()**

```
static INLINE CONST vect_t hadd (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.295.2.47 hadd\_to\_scal()**

```
static INLINE CONST scalar_t hadd_to_scal (
 const vect_t a) [inline], [static]
```

**12.295.3 Field Documentation****12.295.3.1 vect\_size**

```
const constexpr size_t vect_size = 8 [static], [constexpr]
```

**12.295.3.2 alignment**

```
const constexpr size_t alignment = 64 [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [simd512\\_double.inl](#)

**12.296 Simd512\_impl< true, true, false, 8 > Struct Reference**

Inheritance diagram for Simd512\_impl< true, true, false, 8 >:



## Data Structures

- union [Converter](#)

## Public Types

- using `scalar_t` = `uint64_t`
- using `aligned_allocator` = `AlignedAllocator< scalar_t, Alignment(alignment)>`
- using `aligned_vector` = `std::vector< scalar_t, aligned_allocator >`
- template<class `Field` >  
using `is_same_element` = `std::is_same< typename Field::Element, scalar_t >`
- using `simdHalf` = `Simd256< scalar_t >`
- using `vect_t` = `__m512i`
- using `half_t` = `__m256i`

## Static Public Member Functions

- static const `std::string type_string` ()
- static `INLINE CONST vect_t set1` (`const scalar_t x`)
- static `INLINE CONST vect_t set` (`const scalar_t x0, const scalar_t x1, const scalar_t x2, const scalar_t x3, const scalar_t x4, const scalar_t x5, const scalar_t x6, const scalar_t x7`)
- template<class `T` >  
static `INLINE PURE vect_t gather` (`const scalar_t *const p, const T *const idx`)
- static `INLINE PURE vect_t load` (`const scalar_t *const p`)
- static `INLINE PURE vect_t loadu` (`const scalar_t *const p`)
- static `INLINE void store` (`scalar_t *p, vect_t v`)
- template<`uint8_t k`>  
static `INLINE void maskstore` (`scalar_t *p, vect_t v`)
- static `INLINE void storeu` (`scalar_t *p, vect_t v`)
- static `INLINE void stream` (`scalar_t *p, const vect_t v`)
- template<`int s`>  
static `INLINE CONST vect_t sra` (`const vect_t a`)
- static `INLINE CONST vect_t greater` (`vect_t a, vect_t b`)
- static `INLINE CONST vect_t lesser` (`vect_t a, vect_t b`)
- static `INLINE CONST vect_t greater_eq` (`const vect_t a, const vect_t b`)
- static `INLINE CONST vect_t lesser_eq` (`const vect_t a, const vect_t b`)
- static `INLINE CONST vect_t mullo` (`vect_t a, vect_t b`)
- static `INLINE CONST vect_t mulhi` (`vect_t a, vect_t b`)
- static `INLINE CONST vect_t mulx` (`const vect_t a, const vect_t b`)
- static `INLINE CONST vect_t fmaddx` (`const vect_t c, const vect_t a, const vect_t b`)
- static `INLINE vect_t fmaddxin` (`vect_t &c, const vect_t a, const vect_t b`)
- static `INLINE CONST vect_t fnmaddx` (`const vect_t c, const vect_t a, const vect_t b`)
- static `INLINE vect_t fnmaddxin` (`vect_t &c, const vect_t a, const vect_t b`)
- static `INLINE CONST vect_t fmsubx` (`const vect_t c, const vect_t a, const vect_t b`)
- static `INLINE vect_t fmsubxin` (`vect_t &c, const vect_t a, const vect_t b`)
- static `INLINE CONST scalar_t hadd_to_scal` (`const vect_t a`)
- template<class `T` >  
static `constexpr bool valid` (`T *p`)

- `template<class T >`  
`static constexpr bool compliant (T n)`
- `static INLINE CONST vect_t set1 (const scalar_t x)`
- `static INLINE CONST vect_t set (const scalar_t x0, const scalar_t x1, const scalar_t x2, const scalar_t x3, const scalar_t x4, const scalar_t x5, const scalar_t x6, const scalar_t x7)`
- `static INLINE CONST vect_t set (const scalar_t x0, const scalar_t x1, const scalar_t x2, const scalar_t x3)`
- `template<class T >`  
`static INLINE PURE vect_t gather (const scalar_t *const p, const T *const idx)`
- `static INLINE PURE vect_t load (const scalar_t *const p)`
- `static INLINE PURE vect_t loadu (const scalar_t *const p)`
- `static INLINE void store (scalar_t *p, vect_t v)`
- `template<uint8_t k>`  
`static INLINE void maskstore (scalar_t *p, vect_t v)`
- `static INLINE void storeu (scalar_t *p, vect_t v)`
- `static INLINE void stream (scalar_t *p, const vect_t v)`
- `template<int s>`  
`static INLINE CONST vect_t sll (const vect_t a)`
- `template<int s>`  
`static INLINE CONST vect_t srl (const vect_t a)`
- `template<uint8_t s>`  
`static INLINE CONST vect_t shuffle (const vect_t a)`
- `static INLINE CONST vect_t unpacklo_intrinsic (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t unpackhi_intrinsic (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t unpacklo (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t unpackhi (const vect_t a, const vect_t b)`
- `static INLINE void unpacklohi (vect_t &lo, vect_t &hi, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t pack_even (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t pack_odd (const vect_t a, const vect_t b)`
- `static INLINE void pack (vect_t &even, vect_t &odd, const vect_t a, const vect_t b)`
- `static INLINE void transpose (vect_t &r0, vect_t &r1, vect_t &r2, vect_t &r3, vect_t &r4, vect_t &r5, vect_t &r6, vect_t &r7)`
- `template<uint8_t s>`  
`static INLINE CONST vect_t blend (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t add (const vect_t a, const vect_t b)`
- `static INLINE vect_t addin (vect_t &a, const vect_t b)`
- `static INLINE CONST vect_t sub (const vect_t a, const vect_t b)`
- `static INLINE vect_t subin (vect_t &a, const vect_t b)`
- `static INLINE CONST vect_t mul (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmadd (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmaddin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fnmadd (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fnmaddin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t fmsub (const vect_t c, const vect_t a, const vect_t b)`
- `static INLINE vect_t fmsubin (vect_t &c, const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t eq (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t round (const vect_t a)`
- `static INLINE CONST vect_t mask_high ()`
- `static INLINE CONST vect_t mulhi_fast (vect_t x, vect_t y)`
- `static INLINE vect_t mod (vect_t &C, const __m512d &P, const __m512d &INVP, const __m512d &NEGP, const vect_t &POW50REM, const __m512d &MIN, const __m512d &MAX, __m512d &Q, __m512d &T)`
- `static INLINE CONST vect_t zero ()`
- `static INLINE CONST vect_t vor (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t vxor (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t vand (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t vandnot (const vect_t a, const vect_t b)`

**Static Public Attributes**

- `static const constexpr size_t vect_size = 8`
- `static const constexpr size_t alignment = 64`

**Static Protected Member Functions**

- `static INLINE CONST vect_t signbits (const vect_t x)`

**12.296.1 Member Typedef Documentation****12.296.1.1 scalar\_t**

```
using scalar_t = uint64_t
```

**12.296.1.2 aligned\_allocator**

```
using aligned_allocator = AlignedAllocator<scalar_t, Alignment(alignment)>
```

**12.296.1.3 aligned\_vector**

```
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```

**12.296.1.4 is\_same\_element**

```
template<class Field >
using is_same_element = std::is_same<typename Field::Element, scalar_t>
```

**12.296.1.5 simdHalf**

```
using simdHalf = Simd256<scalar_t>
```

**12.296.1.6 vect\_t**

```
using vect_t = __m512i [inherited]
```

**12.296.1.7 half\_t**

```
using half_t = __m256i [inherited]
```

**12.296.2 Member Function Documentation****12.296.2.1 type\_string()**

```
static const std::string type_string () [inline], [static]
```

**12.296.2.2 set1() [1/2]**

```
static INLINE CONST vect_t set1 (
 const scalar_t x) [inline], [static]
```

**12.296.2.3 set() [1/3]**

```
static INLINE CONST vect_t set (
 const scalar_t x0,
 const scalar_t x1,
 const scalar_t x2,
 const scalar_t x3,
 const scalar_t x4,
 const scalar_t x5,
 const scalar_t x6,
 const scalar_t x7) [inline], [static]
```

**12.296.2.4 gather()** [1/2]

```
template<class T >
static INLINE PURE vect_t gather (
 const scalar_t *const p,
 const T *const idx) [inline], [static]
```

**12.296.2.5 load()** [1/2]

```
static INLINE PURE vect_t load (
 const scalar_t *const p) [inline], [static]
```

**12.296.2.6 loadu()** [1/2]

```
static INLINE PURE vect_t loadu (
 const scalar_t *const p) [inline], [static]
```

**12.296.2.7 store()** [1/2]

```
static INLINE void store (
 scalar_t * p,
 vect_t v) [inline], [static]
```

**12.296.2.8 maskstore()** [1/2]

```
template<uint8_t k>
static INLINE void maskstore (
 scalar_t * p,
 vect_t v) [inline], [static]
```

**12.296.2.9 storeu()** [1/2]

```
static INLINE void storeu (
 scalar_t * p,
 vect_t v) [inline], [static]
```

**12.296.2.10 stream()** [1/2]

```
static INLINE void stream (
 scalar_t * p,
 const vect_t v) [inline], [static]
```

**12.296.2.11 sra()**

```
template<int s>
static INLINE CONST vect_t sra (
 const vect_t a) [inline], [static]
```

**12.296.2.12 greater()**

```
static INLINE CONST vect_t greater (
 vect_t a,
 vect_t b) [inline], [static]
```

**12.296.2.13 lesser()**

```
static INLINE CONST vect_t lesser (
 vect_t a,
 vect_t b) [inline], [static]
```



**12.296.2.14 greater\_eq()**

```
static INLINE CONST vect_t greater_eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.296.2.15 lesser\_eq()**

```
static INLINE CONST vect_t lesser_eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.296.2.16 mullo()**

```
static INLINE CONST vect_t mullo (
 vect_t a,
 vect_t b) [inline], [static]
```

**12.296.2.17 mulhi()**

```
static INLINE CONST vect_t mulhi (
 vect_t a,
 vect_t b) [inline], [static]
```

**12.296.2.18 mulx()**

```
static INLINE CONST vect_t mulx (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.296.2.19 fmaddx()**

```
static INLINE CONST vect_t fmaddx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.296.2.20 fmaddxin()**

```
static INLINE vect_t fmaddxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.296.2.21 fnmaddx()**

```
static INLINE CONST vect_t fnmaddx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.296.2.22 fnmaddxin()**

```
static INLINE vect_t fnmaddxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.296.2.23 fmsubx()**

```
static INLINE CONST vect_t fmsubx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.296.2.24 fmsubxin()**

```
static INLINE vect_t fmsubxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.296.2.25 hadd\_to\_scal()**

```
static INLINE CONST scalar_t hadd_to_scal (
 const vect_t a) [inline], [static]
```

**12.296.2.26 valid()**

```
template<class T >
static constexpr bool valid (
 T * p) [inline], [static], [constexpr], [inherited]
```

**12.296.2.27 compliant()**

```
template<class T >
static constexpr bool compliant (
 T n) [inline], [static], [constexpr], [inherited]
```

**12.296.2.28 set1() [2/2]**

```
static INLINE CONST vect_t set1 (
 const scalar_t x) [inline], [static], [inherited]
```

**12.296.2.29 set() [2/3]**

```
static INLINE CONST vect_t set (
 const scalar_t x0,
 const scalar_t x1,
 const scalar_t x2,
 const scalar_t x3,
 const scalar_t x4,
 const scalar_t x5,
 const scalar_t x6,
 const scalar_t x7) [inline], [static], [inherited]
```

**12.296.2.30 set() [3/3]**

```
static INLINE CONST vect_t set (
 const scalar_t x0,
 const scalar_t x1,
 const scalar_t x2,
 const scalar_t x3) [inline], [static], [inherited]
```

**12.296.2.31 gather() [2/2]**

```
template<class T >
static INLINE PURE vect_t gather (
```

```
const scalar_t *const p,
const T *const idx) [inline], [static], [inherited]
```

**12.296.2.32 load() [2/2]**

```
static INLINE PURE vect_t load (
const scalar_t *const p) [inline], [static], [inherited]
```

**12.296.2.33 loadu() [2/2]**

```
static INLINE PURE vect_t loadu (
const scalar_t *const p) [inline], [static], [inherited]
```

**12.296.2.34 store() [2/2]**

```
static INLINE void store (
scalar_t * p,
vect_t v) [inline], [static], [inherited]
```

**12.296.2.35 maskstore() [2/2]**

```
template<uint8_t k>
static INLINE void maskstore (
scalar_t * p,
vect_t v) [inline], [static], [inherited]
```

**12.296.2.36 storeu() [2/2]**

```
static INLINE void storeu (
scalar_t * p,
vect_t v) [inline], [static], [inherited]
```

**12.296.2.37 stream() [2/2]**

```
static INLINE void stream (
scalar_t * p,
const vect_t v) [inline], [static], [inherited]
```

**12.296.2.38 sll()**

```
template<int s>
static INLINE CONST vect_t sll (
const vect_t a) [inline], [static], [inherited]
```

**12.296.2.39 srl()**

```
template<int s>
static INLINE CONST vect_t srl (
const vect_t a) [inline], [static], [inherited]
```

**12.296.2.40 shuffle()**

```
template<uint8_t s>
static INLINE CONST vect_t shuffle (
const vect_t a) [inline], [static], [inherited]
```

**12.296.2.41 unpacklo\_intrinsic()**

```
static INLINE CONST vect_t unpacklo_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.296.2.42 unpackhi\_intrinsic()**

```
static INLINE CONST vect_t unpackhi_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.296.2.43 unpacklo()**

```
static INLINE CONST vect_t unpacklo (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.296.2.44 unpackhi()**

```
static INLINE CONST vect_t unpackhi (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.296.2.45 unpacklohi()**

```
static INLINE void unpacklohi (
 vect_t & lo,
 vect_t & hi,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.296.2.46 pack\_even()**

```
static INLINE CONST vect_t pack_even (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.296.2.47 pack\_odd()**

```
static INLINE CONST vect_t pack_odd (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.296.2.48 pack()**

```
static INLINE void pack (
 vect_t & even,
 vect_t & odd,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.296.2.49 transpose()**

```
static INLINE void transpose (
 vect_t & r0,
 vect_t & r1,
 vect_t & r2,
 vect_t & r3,
 vect_t & r4,
```

```
vect_t & r5,
vect_t & r6,
vect_t & r7) [inline], [static], [inherited]
```

#### 12.296.2.50 blend()

```
template<uint8_t s>
static INLINE CONST vect_t blend (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

#### 12.296.2.51 add()

```
static INLINE CONST vect_t add (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

#### 12.296.2.52 addin()

```
static INLINE vect_t addin (
 vect_t & a,
 const vect_t b) [inline], [static], [inherited]
```

#### 12.296.2.53 sub()

```
static INLINE CONST vect_t sub (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

#### 12.296.2.54 subin()

```
static INLINE vect_t subin (
 vect_t & a,
 const vect_t b) [inline], [static], [inherited]
```

#### 12.296.2.55 mul()

```
static INLINE CONST vect_t mul (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

#### 12.296.2.56 fmadd()

```
static INLINE CONST vect_t fmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

#### 12.296.2.57 fmaddin()

```
static INLINE vect_t fmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

#### 12.296.2.58 fnmadd()

```
static INLINE CONST vect_t fnmadd (
 const vect_t c,
```

```

 const vect_t a,
 const vect_t b) [inline], [static], [inherited]

```

#### 12.296.2.59 fnmaddin()

```

static INLINE vect_t fnmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]

```

#### 12.296.2.60 fmsub()

```

static INLINE CONST vect_t fmsub (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]

```

#### 12.296.2.61 fmsubin()

```

static INLINE vect_t fmsubin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]

```

#### 12.296.2.62 eq()

```

static INLINE CONST vect_t eq (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]

```

#### 12.296.2.63 round()

```

static INLINE CONST vect_t round (
 const vect_t a) [inline], [static], [inherited]

```

#### 12.296.2.64 mask\_high()

```

static INLINE CONST vect_t mask_high () [inline], [static], [inherited]

```

#### 12.296.2.65 mulhi\_fast()

```

INLINE CONST vect_t mulhi_fast (
 vect_t x,
 vect_t y) [static], [inherited]

```

#### 12.296.2.66 mod()

```

INLINE vect_t mod (
 vect_t & C,
 const __m512d & P,
 const __m512d & INV_P,
 const __m512d & NEG_P,
 const vect_t & POW50REM,
 const __m512d & MIN,
 const __m512d & MAX,
 __m512d & Q,
 __m512d & T) [static], [inherited]

```

**12.296.2.67 signbits()**

```
static INLINE CONST vect_t signbits (
 const vect_t x) [inline], [static], [protected], [inherited]
```

**12.296.2.68 zero()**

```
static INLINE CONST vect_t zero () [inline], [static], [inherited]
```

**12.296.2.69 vor()**

```
static INLINE CONST vect_t vor (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.296.2.70 vxor()**

```
static INLINE CONST vect_t vxor (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.296.2.71 vand()**

```
static INLINE CONST vect_t vand (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.296.2.72 vandnot()**

```
static INLINE CONST vect_t vandnot (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.296.3 Field Documentation****12.296.3.1 vect\_size**

```
const constexpr size_t vect_size = 8 [static], [constexpr], [inherited]
```

**12.296.3.2 alignment**

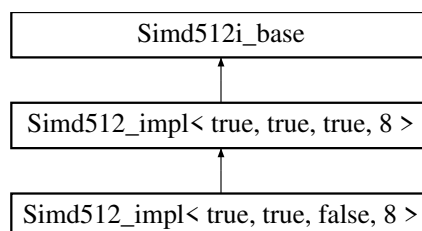
```
const constexpr size_t alignment = 64 [static], [constexpr], [inherited]
```

The documentation for this struct was generated from the following file:

- [simd512\\_int64.inl](#)

**12.297 Simd512\_impl< true, true, true, 8 > Struct Reference**

Inheritance diagram for Simd512\_impl< true, true, true, 8 >:



## Data Structures

- union [Converter](#)

## Public Types

- using vect\_t = \_\_m512i
- using half\_t = \_\_m256i
- using scalar\_t = int64\_t
- using simdHalf = Simd256< scalar\_t >
- using aligned\_allocator = AlignedAllocator< scalar\_t, Alignment(alignment)>
- using aligned\_vector = std::vector< scalar\_t, aligned\_allocator >
- template<class Field >  
using is\_same\_element = std::is\_same< typename Field::Element, scalar\_t >

## Static Public Member Functions

- static const std::string type\_string ()
- template<class T >  
static constexpr bool valid (T \*p)
- template<class T >  
static constexpr bool compliant (T n)
- static INLINE CONST vect\_t set1 (const scalar\_t x)
- static INLINE CONST vect\_t set (const scalar\_t x0, const scalar\_t x1, const scalar\_t x2, const scalar\_t x3, const scalar\_t x4, const scalar\_t x5, const scalar\_t x6, const scalar\_t x7)
- static INLINE CONST vect\_t set (const scalar\_t x0, const scalar\_t x1, const scalar\_t x2, const scalar\_t x3)
- template<class T >  
static INLINE PURE vect\_t gather (const scalar\_t \*const p, const T \*const idx)
- static INLINE PURE vect\_t load (const scalar\_t \*const p)
- static INLINE PURE vect\_t loadu (const scalar\_t \*const p)
- static INLINE void store (scalar\_t \*p, vect\_t v)
- template<uint8\_t k>  
static INLINE void maskstore (scalar\_t \*p, vect\_t v)
- static INLINE void storeu (scalar\_t \*p, vect\_t v)
- static INLINE void stream (scalar\_t \*p, const vect\_t v)
- template<int s>  
static INLINE CONST vect\_t sll (const vect\_t a)
- template<int s>  
static INLINE CONST vect\_t srl (const vect\_t a)
- template<int s>  
static INLINE CONST vect\_t sra (const vect\_t a)
- template<uint8\_t s>  
static INLINE CONST vect\_t shuffle (const vect\_t a)
- static INLINE CONST vect\_t unpacklo\_intrinsic (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t unpackhi\_intrinsic (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t unpacklo (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t unpackhi (const vect\_t a, const vect\_t b)
- static INLINE void unpacklohi (vect\_t &lo, vect\_t &hi, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t pack\_even (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t pack\_odd (const vect\_t a, const vect\_t b)
- static INLINE void pack (vect\_t &even, vect\_t &odd, const vect\_t a, const vect\_t b)
- static INLINE void transpose (vect\_t &r0, vect\_t &r1, vect\_t &r2, vect\_t &r3, vect\_t &r4, vect\_t &r5, vect\_t &r6, vect\_t &r7)
- template<uint8\_t s>  
static INLINE CONST vect\_t blend (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t add (const vect\_t a, const vect\_t b)
- static INLINE vect\_t addin (vect\_t &a, const vect\_t b)



- static INLINE CONST vect\_t sub (const vect\_t a, const vect\_t b)
- static INLINE vect\_t subin (vect\_t &a, const vect\_t b)
- static INLINE CONST vect\_t mullo (vect\_t a, vect\_t b)
- static INLINE CONST vect\_t mul (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t mulhi (vect\_t a, vect\_t b)
- static INLINE CONST vect\_t mulx (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fmadd (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fmaddin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fmaddx (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fmaddxin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fnmadd (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fnmaddin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fnmaddx (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fnmaddxin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fmsub (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fmsubin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t fmsubx (const vect\_t c, const vect\_t a, const vect\_t b)
- static INLINE vect\_t fmsubxin (vect\_t &c, const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t eq (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t greater (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t lesser (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t greater\_eq (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t lesser\_eq (const vect\_t a, const vect\_t b)
- static INLINE CONST scalar\_t hadd\_to\_scal (const vect\_t a)
- static INLINE CONST vect\_t round (const vect\_t a)
- static INLINE CONST vect\_t mask\_high ()
- static INLINE CONST vect\_t mulhi\_fast (vect\_t x, vect\_t y)
- static INLINE vect\_t mod (vect\_t &C, const \_\_m512d &P, const \_\_m512d &INVP, const \_\_m512d &NEGP, const vect\_t &POW50REM, const \_\_m512d &MIN, const \_\_m512d &MAX, \_\_m512d &Q, \_\_m512d &T)
- static INLINE CONST vect\_t zero ()
- static INLINE CONST vect\_t vor (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t vxor (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t vand (const vect\_t a, const vect\_t b)
- static INLINE CONST vect\_t vandnot (const vect\_t a, const vect\_t b)

### Static Public Attributes

- static const constexpr size\_t vect\_size = 8
- static const constexpr size\_t alignment = 64

### Static Protected Member Functions

- static INLINE CONST vect\_t signbits (const vect\_t x)

## 12.297.1 Member Typedef Documentation

### 12.297.1.1 vect\_t

```
using vect_t = __m512i
```

### 12.297.1.2 half\_t

```
using half_t = __m256i
```

### 12.297.1.3 scalar\_t

```
using scalar_t = int64_t
```

**12.297.1.4 simdHalf**

```
using simdHalf = Simd256<scalar_t>
```

**12.297.1.5 aligned\_allocator**

```
using aligned_allocator = AlignedAllocator<scalar_t, Alignment(alignment)>
```

**12.297.1.6 aligned\_vector**

```
using aligned_vector = std::vector<scalar_t, aligned_allocator>
```

**12.297.1.7 is\_same\_element**

```
template<class Field >
using is_same_element = std::is_same<typename Field::Element, scalar_t>
```

**12.297.2 Member Function Documentation****12.297.2.1 type\_string()**

```
static const std::string type_string () [inline], [static]
```

**12.297.2.2 valid()**

```
template<class T >
static constexpr bool valid (
 T * p) [inline], [static], [constexpr]
```

**12.297.2.3 compliant()**

```
template<class T >
static constexpr bool compliant (
 T n) [inline], [static], [constexpr]
```

**12.297.2.4 set1()**

```
static INLINE CONST vect_t set1 (
 const scalar_t x) [inline], [static]
```

**12.297.2.5 set() [1/2]**

```
static INLINE CONST vect_t set (
 const scalar_t x0,
 const scalar_t x1,
 const scalar_t x2,
 const scalar_t x3,
 const scalar_t x4,
 const scalar_t x5,
 const scalar_t x6,
 const scalar_t x7) [inline], [static]
```

**12.297.2.6 set() [2/2]**

```
static INLINE CONST vect_t set (
 const scalar_t x0,
 const scalar_t x1,
 const scalar_t x2,
 const scalar_t x3) [inline], [static]
```

**12.297.2.7 gather()**

```
template<class T >
static INLINE PURE vect_t gather (
 const scalar_t *const p,
 const T *const idx) [inline], [static]
```

**12.297.2.8 load()**

```
static INLINE PURE vect_t load (
 const scalar_t *const p) [inline], [static]
```

**12.297.2.9 loadu()**

```
static INLINE PURE vect_t loadu (
 const scalar_t *const p) [inline], [static]
```

**12.297.2.10 store()**

```
static INLINE void store (
 scalar_t * p,
 vect_t v) [inline], [static]
```

**12.297.2.11 maskstore()**

```
template<uint8_t k>
static INLINE void maskstore (
 scalar_t * p,
 vect_t v) [inline], [static]
```

**12.297.2.12 storeu()**

```
static INLINE void storeu (
 scalar_t * p,
 vect_t v) [inline], [static]
```

**12.297.2.13 stream()**

```
static INLINE void stream (
 scalar_t * p,
 const vect_t v) [inline], [static]
```

**12.297.2.14 sll()**

```
template<int s>
static INLINE CONST vect_t sll (
 const vect_t a) [inline], [static]
```

**12.297.2.15 srl()**

```
template<int s>
static INLINE CONST vect_t srl (
 const vect_t a) [inline], [static]
```

**12.297.2.16 sra()**

```
template<int s>
static INLINE CONST vect_t sra (
 const vect_t a) [inline], [static]
```

**12.297.2.17 shuffle()**

```
template<uint8_t s>
static INLINE CONST vect_t shuffle (
 const vect_t a) [inline], [static]
```

**12.297.2.18 unpacklo\_intrinsic()**

```
static INLINE CONST vect_t unpacklo_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.297.2.19 unpackhi\_intrinsic()**

```
static INLINE CONST vect_t unpackhi_intrinsic (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.297.2.20 unpacklo()**

```
static INLINE CONST vect_t unpacklo (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.297.2.21 unpackhi()**

```
static INLINE CONST vect_t unpackhi (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.297.2.22 unpacklohi()**

```
static INLINE void unpacklohi (
 vect_t & lo,
 vect_t & hi,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.297.2.23 pack\_even()**

```
static INLINE CONST vect_t pack_even (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.297.2.24 pack\_odd()**

```
static INLINE CONST vect_t pack_odd (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.297.2.25 pack()**

```
static INLINE void pack (
 vect_t & even,
 vect_t & odd,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.297.2.26 transpose()**

```
static INLINE void transpose (
 vect_t & r0,
 vect_t & r1,
 vect_t & r2,
 vect_t & r3,
 vect_t & r4,
 vect_t & r5,
 vect_t & r6,
 vect_t & r7) [inline], [static]
```

**12.297.2.27 blend()**

```
template<uint8_t s>
static INLINE CONST vect_t blend (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.297.2.28 add()**

```
static INLINE CONST vect_t add (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.297.2.29 addin()**

```
static INLINE vect_t addin (
 vect_t & a,
 const vect_t b) [inline], [static]
```

**12.297.2.30 sub()**

```
static INLINE CONST vect_t sub (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.297.2.31 subin()**

```
static INLINE vect_t subin (
 vect_t & a,
 const vect_t b) [inline], [static]
```

**12.297.2.32 mullo()**

```
static INLINE CONST vect_t mullo (
 vect_t a,
 vect_t b) [inline], [static]
```

**12.297.2.33 mul()**

```
static INLINE CONST vect_t mul (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.297.2.34 mulhi()**

```
static INLINE CONST vect_t mulhi (
 vect_t a,
 vect_t b) [inline], [static]
```

**12.297.2.35 mulx()**

```
static INLINE CONST vect_t mulx (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.297.2.36 fmadd()**

```
static INLINE CONST vect_t fmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.297.2.37 fmaddin()**

```
static INLINE vect_t fmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.297.2.38 fmaddx()**

```
static INLINE CONST vect_t fmaddx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.297.2.39 fmaddxin()**

```
static INLINE vect_t fmaddxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.297.2.40 fnmadd()**

```
static INLINE CONST vect_t fnmadd (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.297.2.41 fnmaddin()**

```
static INLINE vect_t fnmaddin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.297.2.42 fnmaddx()**

```
static INLINE CONST vect_t fnmaddx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.297.2.43 fnmaddxin()**

```
static INLINE vect_t fnmaddxin (
 vect_t & c,
```

```
const vect_t a,
const vect_t b) [inline], [static]
```

#### 12.297.2.44 fmsub()

```
static INLINE CONST vect_t fmsub (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.297.2.45 fmsubin()

```
static INLINE vect_t fmsubin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.297.2.46 fmsubx()

```
static INLINE CONST vect_t fmsubx (
 const vect_t c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.297.2.47 fmsubxin()

```
static INLINE vect_t fmsubxin (
 vect_t & c,
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.297.2.48 eq()

```
static INLINE CONST vect_t eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.297.2.49 greater()

```
static INLINE CONST vect_t greater (
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.297.2.50 lesser()

```
static INLINE CONST vect_t lesser (
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.297.2.51 greater\_eq()

```
static INLINE CONST vect_t greater_eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

#### 12.297.2.52 lesser\_eq()

```
static INLINE CONST vect_t lesser_eq (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.297.2.53 hadd\_to\_scal()**

```
static INLINE CONST scalar_t hadd_to_scal (
 const vect_t a) [inline], [static]
```

**12.297.2.54 round()**

```
static INLINE CONST vect_t round (
 const vect_t a) [inline], [static]
```

**12.297.2.55 mask\_high()**

```
static INLINE CONST vect_t mask_high () [inline], [static]
```

**12.297.2.56 mulhi\_fast()**

```
INLINE CONST vect_t mulhi_fast (
 vect_t x,
 vect_t y) [static]
```

**12.297.2.57 mod()**

```
INLINE vect_t mod (
 vect_t & C,
 const __m512d & P,
 const __m512d & INVP,
 const __m512d & NEGP,
 const vect_t & POW50REM,
 const __m512d & MIN,
 const __m512d & MAX,
 __m512d & Q,
 __m512d & T) [static]
```

**12.297.2.58 signbits()**

```
static INLINE CONST vect_t signbits (
 const vect_t x) [inline], [static], [protected]
```

**12.297.2.59 zero()**

```
static INLINE CONST vect_t zero () [inline], [static], [inherited]
```

**12.297.2.60 vor()**

```
static INLINE CONST vect_t vor (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.297.2.61 vxor()**

```
static INLINE CONST vect_t vxor (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.297.2.62 vand()**

```
static INLINE CONST vect_t vand (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```



**12.297.2.63 vandnot()**

```
static INLINE CONST vect_t vandnot (
 const vect_t a,
 const vect_t b) [inline], [static], [inherited]
```

**12.297.3 Field Documentation****12.297.3.1 vect\_size**

```
const constexpr size_t vect_size = 8 [static], [constexpr]
```

**12.297.3.2 alignment**

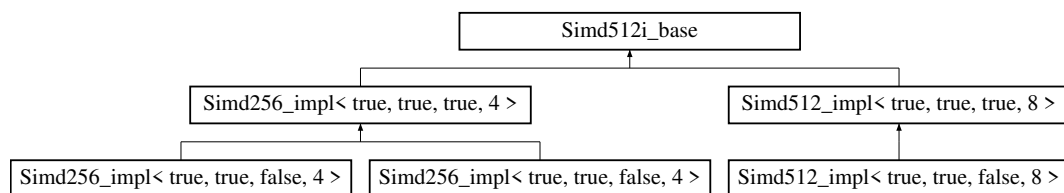
```
const constexpr size_t alignment = 64 [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [simd512\\_int64.inl](#)

**12.298 Simd512i\_base Struct Reference**

Inheritance diagram for Simd512i\_base:

**Public Types**

- `using vect_t = __m512i`

**Static Public Member Functions**

- `static INLINE CONST vect_t zero ()`
- `static INLINE CONST vect_t vor (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t vxor (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t vand (const vect_t a, const vect_t b)`
- `static INLINE CONST vect_t vandnot (const vect_t a, const vect_t b)`

**12.298.1 Member Typedef Documentation****12.298.1.1 vect\_t**

```
using vect_t = __m512i
```

**12.298.2 Member Function Documentation****12.298.2.1 zero()**

```
static INLINE CONST vect_t zero () [inline], [static]
```

**12.298.2.2 vor()**

```
static INLINE CONST vect_t vor (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.298.2.3 vxor()**

```
static INLINE CONST vect_t vxor (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.298.2.4 vand()**

```
static INLINE CONST vect_t vand (
 const vect_t a,
 const vect_t b) [inline], [static]
```

**12.298.2.5 vandnot()**

```
static INLINE CONST vect_t vandnot (
 const vect_t a,
 const vect_t b) [inline], [static]
```

The documentation for this struct was generated from the following file:

- [simd512.inl](#)

**12.299 SimdChooser< T, bool, bool > Struct Template Reference**

```
#include <fflas_simd.h>
```

The documentation for this struct was generated from the following file:

- [fflas\\_simd.h](#)

**12.300 SimdChooser< T, false, b > Struct Template Reference**

```
#include <fflas_simd.h>
```

**Public Types**

- [using value = NoSimd< T >](#)

**12.300.1 Member Typedef Documentation****12.300.1.1 value**

```
template<class T , bool b>
using value = NoSimd<T>
```

The documentation for this struct was generated from the following file:

- [fflas\\_simd.h](#)

**12.301 SimdChooser< T, true, false > Struct Template Reference**

```
#include <fflas_simd.h>
```

**Public Types**

- [using value = NoSimd< T >](#)

### 12.301.1 Member Typedef Documentation

#### 12.301.1.1 value

```
template<class T >
using value = NoSimd<T>
```

The documentation for this struct was generated from the following file:

- [fflas\\_simd.h](#)

## 12.302 SimdChooser< T, true, true > Struct Template Reference

```
#include <fflas_simd.h>
```

### Public Types

- [using value = NoSimd< T >](#)

### 12.302.1 Member Typedef Documentation

#### 12.302.1.1 value

```
template<class T >
using value = NoSimd<T>
```

The documentation for this struct was generated from the following file:

- [fflas\\_simd.h](#)

## 12.303 simdToType< T > Struct Template Reference

The documentation for this struct was generated from the following file:

- [fflas\\_simd.h](#)

## 12.304 Single Struct Reference

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

## 12.305 Sparse< Field, SparseMatrix\_t, IdxT, PtrT > Struct Template Reference

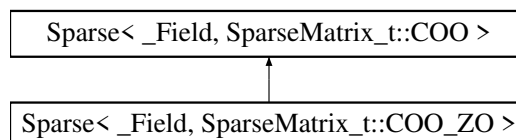
The documentation for this struct was generated from the following file:

- [fflas\\_sparse.h](#)

## 12.306 Sparse< \_Field, SparseMatrix\_t::COO > Struct Template Reference

```
#include <coo.h>
```

Inheritance diagram for Sparse< \_Field, SparseMatrix\_t::COO >:



**Public Types**

- `using Field = _Field`

**Data Fields**

- `index_t* col = nullptr`
- `index_t* row = nullptr`
- `_Field::Element_ptr dat`
- `bool delayed = false`
- `uint64_t kmax = 0`
- `index_t m = 0`
- `index_t n = 0`
- `uint64_t nnz = 0`
- `uint64_t nElements = 0`
- `uint64_t maxrow = 0`

**12.306.1 Member Typedef Documentation****12.306.1.1 Field**

```
template<class _Field >
using Field = _Field
```

**12.306.2 Field Documentation****12.306.2.1 col**

```
template<class _Field >
index_t* col = nullptr
```

**12.306.2.2 row**

```
template<class _Field >
index_t* row = nullptr
```

**12.306.2.3 dat**

```
template<class _Field >
_Field::Element_ptr dat
```

**12.306.2.4 delayed**

```
template<class _Field >
bool delayed = false
```

**12.306.2.5 kmax**

```
template<class _Field >
uint64_t kmax = 0
```

**12.306.2.6 m**

```
template<class _Field >
index_t m = 0
```

**12.306.2.7 n**

```
template<class _Field >
index_t n = 0
```

**12.306.2.8 nnz**

```
template<class _Field >
uint64_t nnz = 0
```

**12.306.2.9 nElements**

```
template<class _Field >
uint64_t nElements = 0
```

**12.306.2.10 maxrow**

```
template<class _Field >
uint64_t maxrow = 0
```

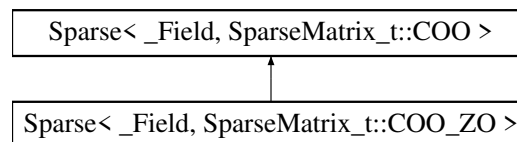
The documentation for this struct was generated from the following file:

- [coo.h](#)

## 12.307 Sparse< \_Field, SparseMatrix\_t::COO\_ZO > Struct Template Reference

```
#include <coo.h>
```

Inheritance diagram for Sparse< \_Field, SparseMatrix\_t::COO\_ZO >:

**Public Types**

- [using Field = \\_Field](#)

**Data Fields**

- `_Field::Element` `cst` = 1
- `index_t` \* `col` = `nullptr`
- `index_t` \* `row` = `nullptr`
- `_Field::Element_ptr` `dat`
- `bool` `delayed` = `false`
- `uint64_t` `kmax` = 0
- `index_t` `m` = 0
- `index_t` `n` = 0
- `uint64_t` `nnz` = 0
- `uint64_t` `nElements` = 0
- `uint64_t` `maxrow` = 0

**12.307.1 Member Typedef Documentation****12.307.1.1 Field**

```
template<class _Field >
using Field = _Field
```

## 12.307.2 Field Documentation

### 12.307.2.1 cst

```
template<class _Field >
_Field::Element cst = 1
```

### 12.307.2.2 col

```
template<class _Field >
index_t* col = nullptr [inherited]
```

### 12.307.2.3 row

```
template<class _Field >
index_t* row = nullptr [inherited]
```

### 12.307.2.4 dat

```
template<class _Field >
_Field::Element_ptr dat [inherited]
```

### 12.307.2.5 delayed

```
template<class _Field >
bool delayed = false [inherited]
```

### 12.307.2.6 kmax

```
template<class _Field >
uint64_t kmax = 0 [inherited]
```

### 12.307.2.7 m

```
template<class _Field >
index_t m = 0 [inherited]
```

### 12.307.2.8 n

```
template<class _Field >
index_t n = 0 [inherited]
```

### 12.307.2.9 nnz

```
template<class _Field >
uint64_t nnz = 0 [inherited]
```

### 12.307.2.10 nElements

```
template<class _Field >
uint64_t nElements = 0 [inherited]
```

### 12.307.2.11 maxrow

```
template<class _Field >
uint64_t maxrow = 0 [inherited]
```

The documentation for this struct was generated from the following file:

- [coo.h](#)

## 12.308 Sparse< \_Field, SparseMatrix\_t::CSR > Struct Template Reference

```
#include <csr.h>
```

Inheritance diagram for Sparse< \_Field, SparseMatrix\_t::CSR >:



### Public Types

- `using Field = _Field`

### Data Fields

- `bool delayed = false`
- `uint64_t kmax = 0`
- `index_t m = 0`
- `index_t n = 0`
- `uint64_t nnz = 0`
- `uint64_t nElements = 0`
- `uint64_t maxrow = 0`
- `index_t * col = nullptr`
- `index_t * st = nullptr`
- `index_t * stend = nullptr`
- `_Field::Element_ptr dat`

## 12.308.1 Member Typedef Documentation

### 12.308.1.1 Field

```
template<class _Field >
using Field = _Field
```

## 12.308.2 Field Documentation

### 12.308.2.1 delayed

```
template<class _Field >
bool delayed = false
```

### 12.308.2.2 kmax

```
template<class _Field >
uint64_t kmax = 0
```

### 12.308.2.3 m

```
template<class _Field >
index_t m = 0
```

### 12.308.2.4 n

```
template<class _Field >
index_t n = 0
```

**12.308.2.5 nnz**

```
template<class _Field >
uint64_t nnz = 0
```

**12.308.2.6 nElements**

```
template<class _Field >
uint64_t nElements = 0
```

**12.308.2.7 maxrow**

```
template<class _Field >
uint64_t maxrow = 0
```

**12.308.2.8 col**

```
template<class _Field >
index_t* col = nullptr
```

**12.308.2.9 st**

```
template<class _Field >
index_t* st = nullptr
```

**12.308.2.10 stend**

```
template<class _Field >
index_t* stend = nullptr
```

**12.308.2.11 dat**

```
template<class _Field >
_Field::Element_ptr dat
```

The documentation for this struct was generated from the following file:

- [csr.h](#)

## 12.309 Sparse<\_Field, SparseMatrix\_t::CSR\_HYB > Struct Template Reference

```
#include <csr_hyb.h>
```

**Public Types**

- [using Field = \\_Field](#)

**Data Fields**

- [bool delayed = false](#)
- [index\\_t \\* col = nullptr](#)
- [index\\_t \\* st = nullptr](#)
- [\\_Field::Element\\_ptr dat](#)
- [uint64\\_t kmax = 0](#)
- [index\\_t m = 0](#)
- [index\\_t n = 0](#)
- [uint64\\_t nnz = 0](#)
- [uint64\\_t nElements = 0](#)



- `uint64_t maxrow` = 0
- `uint64_t nOnes` = 0
- `uint64_t nMOnes` = 0
- `uint64_t nOthers` = 0

## 12.309.1 Member Typedef Documentation

### 12.309.1.1 Field

```
template<class _Field >
using Field = _Field
```

## 12.309.2 Field Documentation

### 12.309.2.1 delayed

```
template<class _Field >
bool delayed = false
```

### 12.309.2.2 col

```
template<class _Field >
index_t* col = nullptr
```

### 12.309.2.3 st

```
template<class _Field >
index_t* st = nullptr
```

### 12.309.2.4 dat

```
template<class _Field >
_Field::Element_ptr dat
```

### 12.309.2.5 kmax

```
template<class _Field >
uint64_t kmax = 0
```

### 12.309.2.6 m

```
template<class _Field >
index_t m = 0
```

### 12.309.2.7 n

```
template<class _Field >
index_t n = 0
```

### 12.309.2.8 nnz

```
template<class _Field >
uint64_t nnz = 0
```

### 12.309.2.9 nElements

```
template<class _Field >
uint64_t nElements = 0
```

**12.309.2.10 maxrow**

```
template<class _Field >
uint64_t maxrow = 0
```

**12.309.2.11 nOnes**

```
template<class _Field >
uint64_t nOnes = 0
```

**12.309.2.12 nMOnes**

```
template<class _Field >
uint64_t nMOnes = 0
```

**12.309.2.13 nOthers**

```
template<class _Field >
uint64_t nOthers = 0
```

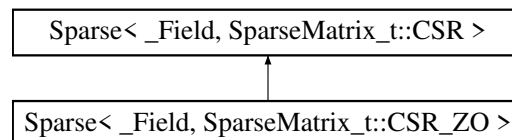
The documentation for this struct was generated from the following file:

- [csr\\_hyb.h](#)

## 12.310 Sparse<\_Field, SparseMatrix\_t::CSR\_ZO > Struct Template Reference

```
#include <csr.h>
```

Inheritance diagram for Sparse<\_Field, SparseMatrix\_t::CSR\_ZO >:

**Public Types**

- [using Field = \\_Field](#)

**Data Fields**

- [int64\\_t cst](#) = 1
- [bool delayed](#) = false
- [uint64\\_t kmax](#) = 0
- [index\\_t m](#) = 0
- [index\\_t n](#) = 0
- [uint64\\_t nnz](#) = 0
- [uint64\\_t nElements](#) = 0
- [uint64\\_t maxrow](#) = 0
- [index\\_t \\* col](#) = nullptr
- [index\\_t \\* st](#) = nullptr
- [index\\_t \\* stend](#) = nullptr
- [\\_Field::Element\\_ptr dat](#)

## 12.310.1 Member Typedef Documentation

### 12.310.1.1 Field

```
template<class _Field >
using Field = _Field
```

## 12.310.2 Field Documentation

### 12.310.2.1 cst

```
template<class _Field >
int64_t cst = 1
```

### 12.310.2.2 delayed

```
template<class _Field >
bool delayed = false
```

### 12.310.2.3 kmax

```
template<class _Field >
uint64_t kmax = 0 [inherited]
```

### 12.310.2.4 m

```
template<class _Field >
index_t m = 0 [inherited]
```

### 12.310.2.5 n

```
template<class _Field >
index_t n = 0 [inherited]
```

### 12.310.2.6 nnz

```
template<class _Field >
uint64_t nnz = 0 [inherited]
```

### 12.310.2.7 nElements

```
template<class _Field >
uint64_t nElements = 0 [inherited]
```

### 12.310.2.8 maxrow

```
template<class _Field >
uint64_t maxrow = 0 [inherited]
```

### 12.310.2.9 col

```
template<class _Field >
index_t* col = nullptr [inherited]
```

### 12.310.2.10 st

```
template<class _Field >
index_t* st = nullptr [inherited]
```

**12.310.2.11 stend**

```
template<class _Field >
index_t* stend = nullptr [inherited]
```

**12.310.2.12 dat**

```
template<class _Field >
_Field::Element_ptr dat [inherited]
```

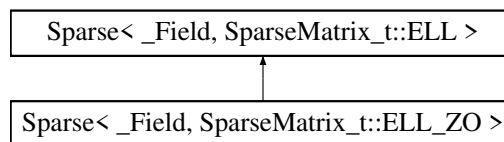
The documentation for this struct was generated from the following file:

- [csr.h](#)

## 12.311 Sparse< \_Field, SparseMatrix\_t::ELL > Struct Template Reference

```
#include <ell.h>
```

Inheritance diagram for Sparse< \_Field, SparseMatrix\_t::ELL >:

**Public Types**

- [using Field = \\_Field](#)

**Data Fields**

- [bool delayed = false](#)
- [uint64\\_t kmax = 0](#)
- [index\\_t m = 0](#)
- [index\\_t n = 0](#)
- [index\\_t ld = 0](#)
- [uint64\\_t nnz = 0](#)
- [uint64\\_t nElements = 0](#)
- [uint64\\_t maxrow = 0](#)
- [index\\_t \\* col = nullptr](#)
- [\\_Field::Element\\_ptr dat](#)

**12.311.1 Member Typedef Documentation****12.311.1.1 Field**

```
template<class _Field >
using Field = _Field
```

**12.311.2 Field Documentation****12.311.2.1 delayed**

```
template<class _Field >
bool delayed = false
```

**12.311.2.2 kmax**

```
template<class _Field >
uint64_t kmax = 0
```

**12.311.2.3 m**

```
template<class _Field >
index_t m = 0
```

**12.311.2.4 n**

```
template<class _Field >
index_t n = 0
```

**12.311.2.5 ld**

```
template<class _Field >
index_t ld = 0
```

**12.311.2.6 nnz**

```
template<class _Field >
uint64_t nnz = 0
```

**12.311.2.7 nElements**

```
template<class _Field >
uint64_t nElements = 0
```

**12.311.2.8 maxrow**

```
template<class _Field >
uint64_t maxrow = 0
```

**12.311.2.9 col**

```
template<class _Field >
index_t* col = nullptr
```

**12.311.2.10 dat**

```
template<class _Field >
_Field::Element_ptr dat
```

The documentation for this struct was generated from the following file:

- [ell.h](#)

## 12.312 Sparse< \_Field, SparseMatrix\_t::ELL\_simd > Struct Template Reference

```
#include <ell_simd.h>
```

Inheritance diagram for Sparse< \_Field, SparseMatrix\_t::ELL\_simd >:



## Data Fields

- `bool delayed = false`
- `int chunk = 0`
- `index_t m = 0`
- `index_t n = 0`
- `index_t ld = 0`
- `uint64_t kmax = 0`
- `uint64_t nnz = 0`
- `uint64_t nElements = 0`
- `uint64_t maxrow = 0`
- `uint64_t nChunks = 0`
- `index_t * col = nullptr`
- `_Field::Element_ptr dat`

## 12.312.1 Field Documentation

### 12.312.1.1 delayed

```
template<class _Field >
bool delayed = false
```

### 12.312.1.2 chunk

```
template<class _Field >
int chunk = 0
```

### 12.312.1.3 m

```
template<class _Field >
index_t m = 0
```

### 12.312.1.4 n

```
template<class _Field >
index_t n = 0
```

### 12.312.1.5 ld

```
template<class _Field >
index_t ld = 0
```

### 12.312.1.6 kmax

```
template<class _Field >
uint64_t kmax = 0
```

### 12.312.1.7 nnz

```
template<class _Field >
uint64_t nnz = 0
```

### 12.312.1.8 nElements

```
template<class _Field >
uint64_t nElements = 0
```

**12.312.1.9 maxrow**

```
template<class _Field >
uint64_t maxrow = 0
```

**12.312.1.10 nChunks**

```
template<class _Field >
uint64_t nChunks = 0
```

**12.312.1.11 col**

```
template<class _Field >
index_t* col = nullptr
```

**12.312.1.12 dat**

```
template<class _Field >
_Field::Element_ptr dat
```

The documentation for this struct was generated from the following file:

- [ell\\_simd.h](#)

## 12.313 Sparse< \_Field, SparseMatrix\_t::ELL\_simd\_ZO > Struct Template Reference

```
#include <ell_simd.h>
```

Inheritance diagram for Sparse< \_Field, SparseMatrix\_t::ELL\_simd\_ZO >:

**Data Fields**

- \_Field::Element [cst](#) = 1
- [bool](#) [delayed](#) = [false](#)
- [int](#) [chunk](#) = 0
- [index\\_t](#) [m](#) = 0
- [index\\_t](#) [n](#) = 0
- [index\\_t](#) [ld](#) = 0
- [uint64\\_t](#) [kmax](#) = 0
- [uint64\\_t](#) [nnz](#) = 0
- [uint64\\_t](#) [nElements](#) = 0
- [uint64\\_t](#) [maxrow](#) = 0
- [uint64\\_t](#) [nChunks](#) = 0
- [index\\_t](#) \* [col](#) = [nullptr](#)
- \_Field::Element\_ptr [dat](#)

**12.313.1 Field Documentation****12.313.1.1 cst**

```
template<class _Field >
_Field::Element cst = 1
```

#### 12.313.1.2 delayed

```
template<class _Field >
bool delayed = false [inherited]
```

#### 12.313.1.3 chunk

```
template<class _Field >
int chunk = 0 [inherited]
```

#### 12.313.1.4 m

```
template<class _Field >
index_t m = 0 [inherited]
```

#### 12.313.1.5 n

```
template<class _Field >
index_t n = 0 [inherited]
```

#### 12.313.1.6 ld

```
template<class _Field >
index_t ld = 0 [inherited]
```

#### 12.313.1.7 kmax

```
template<class _Field >
uint64_t kmax = 0 [inherited]
```

#### 12.313.1.8 nnz

```
template<class _Field >
uint64_t nnz = 0 [inherited]
```

#### 12.313.1.9 nElements

```
template<class _Field >
uint64_t nElements = 0 [inherited]
```

#### 12.313.1.10 maxrow

```
template<class _Field >
uint64_t maxrow = 0 [inherited]
```

#### 12.313.1.11 nChunks

```
template<class _Field >
uint64_t nChunks = 0 [inherited]
```

#### 12.313.1.12 col

```
template<class _Field >
index_t* col = nullptr [inherited]
```

#### 12.313.1.13 dat

```
template<class _Field >
_Field::Element_ptr dat [inherited]
```

The documentation for this struct was generated from the following file:

- [ell\\_simd.h](#)



## 12.314 Sparse< \_Field, SparseMatrix\_t::ELL\_ZO > Struct Template Reference

```
#include <ell.h>
```

Inheritance diagram for Sparse< \_Field, SparseMatrix\_t::ELL\_ZO >:



### Public Types

- `using Field = _Field`

### Data Fields

- `_Field::Element cst = 1`
- `bool delayed = false`
- `uint64_t kmax = 0`
- `index_t m = 0`
- `index_t n = 0`
- `index_t ld = 0`
- `uint64_t nnz = 0`
- `uint64_t nElements = 0`
- `uint64_t maxrow = 0`
- `index_t * col = nullptr`
- `_Field::Element_ptr dat`

### 12.314.1 Member Typedef Documentation

#### 12.314.1.1 Field

```
template<class _Field >
using Field = _Field
```

### 12.314.2 Field Documentation

#### 12.314.2.1 cst

```
template<class _Field >
_Field::Element cst = 1
```

#### 12.314.2.2 delayed

```
template<class _Field >
bool delayed = false [inherited]
```

#### 12.314.2.3 kmax

```
template<class _Field >
uint64_t kmax = 0 [inherited]
```

#### 12.314.2.4 m

```
template<class _Field >
index_t m = 0 [inherited]
```

**12.314.2.5 n**

```
template<class _Field >
index_t n = 0 [inherited]
```

**12.314.2.6 ld**

```
template<class _Field >
index_t ld = 0 [inherited]
```

**12.314.2.7 nnz**

```
template<class _Field >
uint64_t nnz = 0 [inherited]
```

**12.314.2.8 nElements**

```
template<class _Field >
uint64_t nElements = 0 [inherited]
```

**12.314.2.9 maxrow**

```
template<class _Field >
uint64_t maxrow = 0 [inherited]
```

**12.314.2.10 col**

```
template<class _Field >
index_t* col = nullptr [inherited]
```

**12.314.2.11 dat**

```
template<class _Field >
_Field::Element_ptr dat [inherited]
```

The documentation for this struct was generated from the following file:

- [ell.h](#)

## 12.315 Sparse<\_Field, SparseMatrix\_t::HYB\_ZO > Struct Template Reference

```
#include <hyb_zo.h>
```

**Public Types**

- [using Field = \\_Field](#)
- [typedef Sparse<\\_Field, SparseMatrix\\_t::HYB\\_ZO > Self\\_t](#)

**Data Fields**

- [bool delayed = false](#)
- [uint64\\_t kmax = 0](#)
- [index\\_t m = 0](#)
- [index\\_t n = 0](#)
- [uint64\\_t nnz = 0](#)
- [uint64\\_t maxrow = 0](#)
- [uint64\\_t nElements = 0](#)
- [Sparse<\\_Field, SparseMatrix\\_t::CSR > \\* dat = nullptr](#)
- [Sparse<\\_Field, SparseMatrix\\_t::CSR\\_ZO > \\* one = nullptr](#)
- [Sparse<\\_Field, SparseMatrix\\_t::CSR\\_ZO > \\* mone = nullptr](#)

## 12.315.1 Member Typedef Documentation

### 12.315.1.1 Field

```
template<class _Field >
using Field = _Field
```

### 12.315.1.2 Self\_t

```
template<class _Field >
typedef Sparse<_Field, SparseMatrix_t::HYB_ZO> Self_t
```

## 12.315.2 Field Documentation

### 12.315.2.1 delayed

```
template<class _Field >
bool delayed = false
```

### 12.315.2.2 kmax

```
template<class _Field >
uint64_t kmax = 0
```

### 12.315.2.3 m

```
template<class _Field >
index_t m = 0
```

### 12.315.2.4 n

```
template<class _Field >
index_t n = 0
```

### 12.315.2.5 nnz

```
template<class _Field >
uint64_t nnz = 0
```

### 12.315.2.6 maxrow

```
template<class _Field >
uint64_t maxrow = 0
```

### 12.315.2.7 nElements

```
template<class _Field >
uint64_t nElements = 0
```

### 12.315.2.8 dat

```
template<class _Field >
Sparse<_Field, SparseMatrix_t::CSR>* dat = nullptr
```

### 12.315.2.9 one

```
template<class _Field >
Sparse<_Field, SparseMatrix_t::CSR_ZO>* one = nullptr
```

### 12.315.2.10 mone

```
template<class _Field >
Sparse<_Field, SparseMatrix_t::CSR_ZO>* mone = nullptr
```

The documentation for this struct was generated from the following file:

- [hyb\\_zo.h](#)

## 12.316 Sparse< \_Field, SparseMatrix\_t::SELL > Struct Template Reference

```
#include <sell.h>
Inheritance diagram for Sparse< _Field, SparseMatrix_t::SELL >:
```



### Public Types

- [using Field = \\_Field](#)

### Data Fields

- [bool delayed = false](#)
- [int chunk = 0](#)
- [index\\_t kmax = 0](#)
- [index\\_t m = 0](#)
- [index\\_t n = 0](#)
- [index\\_t maxrow = 0](#)
- [index\\_t sigma = 0](#)
- [index\\_t nChunks = 0](#)
- [uint64\\_t nnz = 0](#)
- [uint64\\_t nElements = 0](#)
- [index\\_t \\* perm = nullptr](#)
- [uint64\\_t \\* st = nullptr](#)
- [index\\_t \\* chunkSize = nullptr](#)
- [index\\_t \\* col = nullptr](#)
- [\\_Field::Element\\_ptr dat](#)

## 12.316.1 Member Typedef Documentation

### 12.316.1.1 Field

```
template<class _Field >
using Field = _Field
```

## 12.316.2 Field Documentation

### 12.316.2.1 delayed

```
template<class _Field >
bool delayed = false
```

#### 12.316.2.2 chunk

```
template<class _Field >
int chunk = 0
```

#### 12.316.2.3 kmax

```
template<class _Field >
index_t kmax = 0
```

#### 12.316.2.4 m

```
template<class _Field >
index_t m = 0
```

#### 12.316.2.5 n

```
template<class _Field >
index_t n = 0
```

#### 12.316.2.6 maxrow

```
template<class _Field >
index_t maxrow = 0
```

#### 12.316.2.7 sigma

```
template<class _Field >
index_t sigma = 0
```

#### 12.316.2.8 nChunks

```
template<class _Field >
index_t nChunks = 0
```

#### 12.316.2.9 nnz

```
template<class _Field >
uint64_t nnz = 0
```

#### 12.316.2.10 nElements

```
template<class _Field >
uint64_t nElements = 0
```

#### 12.316.2.11 perm

```
template<class _Field >
index_t* perm = nullptr
```

#### 12.316.2.12 st

```
template<class _Field >
uint64_t* st = nullptr
```

#### 12.316.2.13 chunkSize

```
template<class _Field >
index_t* chunkSize = nullptr
```

**12.316.2.14 col**

```
template<class _Field >
index_t* col = nullptr
```

**12.316.2.15 dat**

```
template<class _Field >
_Field::Element_ptr dat
```

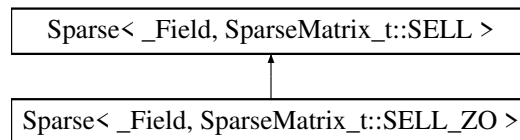
The documentation for this struct was generated from the following file:

- [sell.h](#)

## 12.317 Sparse< \_Field, SparseMatrix\_t::SELL\_ZO > Struct Template Reference

```
#include <sell.h>
```

Inheritance diagram for Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >:

**Public Types**

- [using Field = \\_Field](#)

**Data Fields**

- `_Field::Element` [cst](#) = 1
- [bool](#) [delayed](#) = [false](#)
- [int](#) [chunk](#) = 0
- [index\\_t](#) [kmax](#) = 0
- [index\\_t](#) [m](#) = 0
- [index\\_t](#) [n](#) = 0
- [index\\_t](#) [maxrow](#) = 0
- [index\\_t](#) [sigma](#) = 0
- [index\\_t](#) [nChunks](#) = 0
- [uint64\\_t](#) [nnz](#) = 0
- [uint64\\_t](#) [nElements](#) = 0
- [index\\_t](#) \* [perm](#) = [nullptr](#)
- [uint64\\_t](#) \* [st](#) = [nullptr](#)
- [index\\_t](#) \* [chunkSize](#) = [nullptr](#)
- [index\\_t](#) \* [col](#) = [nullptr](#)
- `_Field::Element_ptr` [dat](#)

**12.317.1 Member Typedef Documentation****12.317.1.1 Field**

```
template<class _Field >
using Field = _Field
```

## 12.317.2 Field Documentation

### 12.317.2.1 cst

```
template<class _Field >
_Field::Element cst = 1
```

### 12.317.2.2 delayed

```
template<class _Field >
bool delayed = false [inherited]
```

### 12.317.2.3 chunk

```
template<class _Field >
int chunk = 0 [inherited]
```

### 12.317.2.4 kmax

```
template<class _Field >
index_t kmax = 0 [inherited]
```

### 12.317.2.5 m

```
template<class _Field >
index_t m = 0 [inherited]
```

### 12.317.2.6 n

```
template<class _Field >
index_t n = 0 [inherited]
```

### 12.317.2.7 maxrow

```
template<class _Field >
index_t maxrow = 0 [inherited]
```

### 12.317.2.8 sigma

```
template<class _Field >
index_t sigma = 0 [inherited]
```

### 12.317.2.9 nChunks

```
template<class _Field >
index_t nChunks = 0 [inherited]
```

### 12.317.2.10 nnz

```
template<class _Field >
uint64_t nnz = 0 [inherited]
```

### 12.317.2.11 nElements

```
template<class _Field >
uint64_t nElements = 0 [inherited]
```

### 12.317.2.12 perm

```
template<class _Field >
index_t* perm = nullptr [inherited]
```

**12.317.2.13 st**

```
template<class _Field >
uint64_t* st = nullptr [inherited]
```

**12.317.2.14 chunkSize**

```
template<class _Field >
index_t* chunkSize = nullptr [inherited]
```

**12.317.2.15 col**

```
template<class _Field >
index_t* col = nullptr [inherited]
```

**12.317.2.16 dat**

```
template<class _Field >
_Field::Element_ptr dat [inherited]
```

The documentation for this struct was generated from the following file:

- [sell.h](#)

**12.318 SpMat< Field, flag > Struct Template Reference**

```
#include <fflas_sparse.h>
```

**Data Fields**

- [FFLAS::CooMat< Field > \\* \\_coo = nullptr](#)
- [FFLAS::CsrMat< Field > \\* \\_csr = nullptr](#)
- [FFLAS::EllMat< Field > \\* \\_ell = nullptr](#)

**12.318.1 Field Documentation****12.318.1.1 \_coo**

```
template<class Field , int flag = HelperFlag::none>
FFLAS::CooMat<Field>* _coo = nullptr
```

**12.318.1.2 \_csr**

```
template<class Field , int flag = HelperFlag::none>
FFLAS::CsrMat<Field>* _csr = nullptr
```

**12.318.1.3 \_ell**

```
template<class Field , int flag = HelperFlag::none>
FFLAS::EllMat<Field>* _ell = nullptr
```

The documentation for this struct was generated from the following file:

- [fflas\\_sparse.h](#)

**12.319 StatsMatrix Struct Reference**

```
#include <utils.h>
```



## Data Fields

- `uint64_t rowdim` = 0
- `uint64_t coldim` = 0
- `uint64_t nOnes` = 0
- `uint64_t nMOnes` = 0
- `uint64_t nOthers` = 0
- `uint64_t nnz` = 0
- `uint64_t maxRow` = 0
- `uint64_t minRow` = 0
- `uint64_t averageRow` = 0
- `uint64_t deviationRow` = 0
- `uint64_t maxCol` = 0
- `uint64_t minCol` = 0
- `uint64_t averageCol` = 0
- `uint64_t deviationCol` = 0
- `uint64_t minColDifference` = 0
- `uint64_t maxColDifference` = 0
- `uint64_t averageColDifference` = 0
- `uint64_t deviationColDifference` = 0
- `uint64_t minRowDifference` = 0
- `uint64_t maxRowDifference` = 0
- `uint64_t averageRowDifference` = 0
- `uint64_t deviationRowDifference` = 0
- `uint64_t nDenseRows` = 0
- `uint64_t nDenseCols` = 0
- `uint64_t nEmptyRows` = 0
- `uint64_t nEmptyCols` = 0
- `uint64_t nEmptyColsEnd` = 0
- `std::vector< uint64_t > denseRows`
- `std::vector< uint64_t > denseCols`

## 12.319.1 Field Documentation

### 12.319.1.1 rowdim

`uint64_t rowdim` = 0

### 12.319.1.2 coldim

`uint64_t coldim` = 0

### 12.319.1.3 nOnes

`uint64_t nOnes` = 0

### 12.319.1.4 nMOnes

`uint64_t nMOnes` = 0

### 12.319.1.5 nOthers

`uint64_t nOthers` = 0

### 12.319.1.6 nnz

`uint64_t nnz` = 0

**12.319.1.7 maxRow**

```
uint64_t maxRow = 0
```

**12.319.1.8 minRow**

```
uint64_t minRow = 0
```

**12.319.1.9 averageRow**

```
uint64_t averageRow = 0
```

**12.319.1.10 deviationRow**

```
uint64_t deviationRow = 0
```

**12.319.1.11 maxCol**

```
uint64_t maxCol = 0
```

**12.319.1.12 minCol**

```
uint64_t minCol = 0
```

**12.319.1.13 averageCol**

```
uint64_t averageCol = 0
```

**12.319.1.14 deviationCol**

```
uint64_t deviationCol = 0
```

**12.319.1.15 minColDifference**

```
uint64_t minColDifference = 0
```

**12.319.1.16 maxColDifference**

```
uint64_t maxColDifference = 0
```

**12.319.1.17 averageColDifference**

```
uint64_t averageColDifference = 0
```

**12.319.1.18 deviationColDifference**

```
uint64_t deviationColDifference = 0
```

**12.319.1.19 minRowDifference**

```
uint64_t minRowDifference = 0
```

**12.319.1.20 maxRowDifference**

```
uint64_t maxRowDifference = 0
```

**12.319.1.21 averageRowDifference**

```
uint64_t averageRowDifference = 0
```

**12.319.1.22 deviationRowDifference**

```
uint64_t deviationRowDifference = 0
```

**12.319.1.23 nDenseRows**

```
uint64_t nDenseRows = 0
```

**12.319.1.24 nDenseCols**

```
uint64_t nDenseCols = 0
```

**12.319.1.25 nEmptyRows**

```
uint64_t nEmptyRows = 0
```

**12.319.1.26 nEmptyCols**

```
uint64_t nEmptyCols = 0
```

**12.319.1.27 nEmptyColsEnd**

```
uint64_t nEmptyColsEnd = 0
```

**12.319.1.28 denseRows**

```
std::vector<uint64_t> denseRows
```

**12.319.1.29 denseCols**

```
std::vector<uint64_t> denseCols
```

The documentation for this struct was generated from the following file:

- [utils.h](#)

**12.320 support\_fast\_mod< T > Struct Template Reference**

```
#include <fflas_freduce.h>
```

Inheritance diagram for support\_fast\_mod< T >:



The documentation for this struct was generated from the following file:

- [fflas\\_freduce.h](#)

**12.321 support\_fast\_mod< double > Struct Reference**

```
#include <fflas_freduce.h>
```

Inheritance diagram for support\_fast\_mod< double >:



The documentation for this struct was generated from the following file:

- [fflas\\_freduce.h](#)

### 12.322 support\_fast\_mod< float > Struct Reference

```
#include <fflas_freduce.h>
```

Inheritance diagram for support\_fast\_mod< float >:



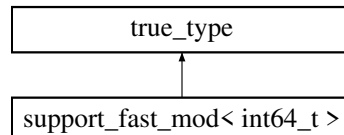
The documentation for this struct was generated from the following file:

- [fflas\\_freduce.h](#)

### 12.323 support\_fast\_mod< int64\_t > Struct Reference

```
#include <fflas_freduce.h>
```

Inheritance diagram for support\_fast\_mod< int64\_t >:



The documentation for this struct was generated from the following file:

- [fflas\\_freduce.h](#)

### 12.324 support\_simd< T > Struct Template Reference

```
#include <fflas_simd.h>
```

Inheritance diagram for support\_simd< T >:



The documentation for this struct was generated from the following file:

- [fflas\\_simd.h](#)

## 12.325 support\_simd\_add< T > Struct Template Reference

```
#include <fflas_fadd.h>
```

Inheritance diagram for support\_simd\_add< T >:



The documentation for this struct was generated from the following file:

- [fflas\\_fadd.h](#)

## 12.326 support\_simd\_mod< T > Struct Template Reference

```
#include <fflas_freduce.h>
```

Inheritance diagram for support\_simd\_mod< T >:



The documentation for this struct was generated from the following file:

- [fflas\\_freduce.h](#)

## 12.327 Test< Elt > Class Template Reference

### Public Types

- [using Field](#) = [Modular](#)< [Elt](#) >
- [using Elt\\_ptr](#) = [typename](#) [Field](#)::[Element\\_ptr](#)
- [using Residu](#) = [typename](#) [Field](#)::[Residu\\_t](#)
- [template](#)<[bool](#) B, [class](#) T = void>  
[using enable\\_if\\_t](#) = [typename](#) [std::enable\\_if](#)< B, T >::type
- [template](#)<[typename](#) Simd >  
[using is\\_same\\_element](#) = [typename](#) [Simd](#)::[template is\\_same\\_element](#)< [Field](#) >
- [template](#)<[typename](#) E >  
[using enable\\_if\\_no\\_simd\\_t](#) = [enable\\_if\\_t](#)< [Simd](#)< E >::vect\_size==1 >
- [template](#)<[typename](#) E >  
[using enable\\_if\\_simd128\\_t](#) = [enable\\_if\\_t](#)< [sizeof](#)(E) \*[Simd](#)< E >::vect\_size==16 >
- [template](#)<[typename](#) E >  
[using enable\\_if\\_simd256\\_t](#) = [enable\\_if\\_t](#)< [sizeof](#)(E) \*[Simd](#)< E >::vect\_size==32 >
- [template](#)<[typename](#) E >  
[using enable\\_if\\_simd512\\_t](#) = [enable\\_if\\_t](#)< [sizeof](#)(E) \*[Simd](#)< E >::vect\_size==64 >

### Public Member Functions

- [Test](#) ([size\\_t](#) mm, [size\\_t](#) nn)
- [template](#)<[typename](#) Simd = NoSimd<Elt>, [enable\\_if\\_t](#)< [is\\_same\\_element](#)< Simd >::value > \* = nullptr>  
[bool test\\_ftranspose](#) ([size\\_t](#) m, [size\\_t](#) n, [Elt\\_ptr](#) A, [size\\_t](#) lda, [Elt\\_ptr](#) B, [size\\_t](#) ldb)

- `template<typename Simd = NoSimd<Elt>, enable_if_t<is_same_element< Simd >::value > * = nullptr>  
bool doTests ()`
- `template<typename _E = Elt, enable_if_t<is_same< _E, Elt >::value > * = nullptr, enable_if_no_simd_t< _E > * = nullptr>  
bool run ()`

### Static Public Member Functions

- `template<typename _E = Elt, enable_if_t<!is_same< _E, Givaro::Integer >::value > * = nullptr>  
static Residu cardinality ()`
- `template<typename _E = Elt, enable_if_t<is_same< _E, Givaro::Integer >::value > * = nullptr>  
static Residu cardinality ()`

### Protected Attributes

- `Field F`
- `size_t _mm`
- `size_t _nn`

## 12.327.1 Member Typedef Documentation

### 12.327.1.1 Field

```
template<typename Elt >
using Field = Modular<Elt>
```

### 12.327.1.2 Elt\_ptr

```
template<typename Elt >
using Elt_ptr = typename Field::Element_ptr
```

### 12.327.1.3 Residu

```
template<typename Elt >
using Residu = typename Field::Residu_t
```

### 12.327.1.4 enable\_if\_t

```
template<typename Elt >
template<bool B, class T = void>
using enable_if_t = typename std::enable_if<B, T>::type
```

### 12.327.1.5 is\_same\_element

```
template<typename Elt >
template<typename Simd >
using is_same_element = typename Simd::template is_same_element<Field>
```

### 12.327.1.6 enable\_if\_no\_simd\_t

```
template<typename Elt >
template<typename E >
using enable_if_no_simd_t = enable_if_t<Simd<E>::vect_size == 1>
```

### 12.327.1.7 enable\_if\_simd128\_t

```
template<typename Elt >
template<typename E >
using enable_if_simd128_t = enable_if_t<sizeof(E)*Simd<E>::vect_size == 16>
```

**12.327.1.8 enable\_if\_simd256\_t**

```
template<typename Elt >
template<typename E >
using enable_if_simd256_t = enable_if_t<sizeof(E)*Simd<E>::vect_size == 32>
```

**12.327.1.9 enable\_if\_simd512\_t**

```
template<typename Elt >
template<typename E >
using enable_if_simd512_t = enable_if_t<sizeof(E)*Simd<E>::vect_size == 64>
```

**12.327.2 Constructor & Destructor Documentation****12.327.2.1 Test()**

```
template<typename Elt >
Test (
 size_t mm,
 size_t nn) [inline]
```

**12.327.3 Member Function Documentation****12.327.3.1 cardinality() [1/2]**

```
template<typename Elt >
template<typename _E = Elt, enable_if_t<!is_same< _E, Givaro::Integer >::value > * = nullptr>
static Residu cardinality () [inline], [static]
```

**12.327.3.2 cardinality() [2/2]**

```
template<typename Elt >
template<typename _E = Elt, enable_if_t< is_same< _E, Givaro::Integer >::value > * = nullptr>
static Residu cardinality () [inline], [static]
```

**12.327.3.3 test\_ftranspose()**

```
template<typename Elt >
template<typename Simd = NoSimd<Elt>, enable_if_t< is_same_element< Simd >::value > * =
nullptr>
bool test_ftranspose (
 size_t m,
 size_t n,
 Elt_ptr A,
 size_t lda,
 Elt_ptr B,
 size_t ldb) [inline]
```

**12.327.3.4 doTests()**

```
template<typename Elt >
template<typename Simd = NoSimd<Elt>, enable_if_t< is_same_element< Simd >::value > * =
nullptr>
bool doTests () [inline]
```

**12.327.3.5 run()**

```
template<typename Elt >
template<typename _E = Elt, enable_if_t< is_same< _E, Elt >::value > * = nullptr, enable_if_no_simd_t<
_E > * = nullptr>
bool run () [inline]
```

## 12.327.4 Field Documentation

### 12.327.4.1 F

```
template<typename Elt >
Field F [protected]
```

### 12.327.4.2 \_mm

```
template<typename Elt >
size_t _mm [protected]
```

### 12.327.4.3 \_nn

```
template<typename Elt >
size_t _nn [protected]
```

The documentation for this class was generated from the following file:

- [test-storage-transpose.C](#)

## 12.328 TestOneMethod< Simd > Class Template Reference

### Public Types

- `using Element = typename Simd::scalar_t`
- `using vect_t = typename Simd::vect_t`
- `using vectElt = vector< Element >`
- `template<bool B, typename T = void>`  
`using enable_if_t = typename enable_if< B, T >::type`

### Public Member Functions

- `template<typename... AScal, typename RScal, typename... ASimd, typename RSimd, enable_if_t< sizeof...(AScal)==sizeof...(ASimd)>`  
`* = nullptr, enable_if_t< count_nonconst_lvalue_reference< AScal... >::n==count_nonconst_lvalue_reference< ASimd... >::n > *`  
`= nullptr, enable_if_t< is_all_same< AScal... >::value > * = nullptr, enable_if_t< is_all_same< vect_t, ASimd... >::value > * =`  
`nullptr>`  
`TestOneMethod (function< RSimd(ASimd...)> fsimd, function< RScal(AScal...)> fscal, function<`  
`void(vector< vectElt > &)> genInputs, string fname)`
- `template<typename Ret, typename... AScal>`  
`enable_if_t< is_all_same< Element, AScal... >::value &&std::is_convertible< Ret, Element >::value, void`  
`> evaluate_scalar_method (function< Ret(AScal...)> fscal)`
- `template<typename... AScal>`  
`enable_if_t< is_all_same< vectElt, AScal... >::value, void > evaluate_scalar_method (function<`  
`vectElt(AScal...)> fscal)`
- `template<typename... AScal>`  
`enable_if_t< is_all_same< vectElt, AScal... >::value, void > evaluate_scalar_method (function<`  
`void(AScal...)> fscal)`
- `template<typename Ret, typename... ASimd>`  
`enable_if_t< is_all_same< vect_t, ASimd... >::value &&std::is_convertible< Ret, vect_t >::value, void >`  
`evaluate_simd_method (function< Ret(ASimd...)> fsimd, array< vect_t, sizeof...(ASimd)> &simd_in)`
- `template<typename... ASimd>`  
`enable_if_t< is_all_same< vect_t, ASimd... >::value, void > evaluate_simd_method (function<`  
`void(ASimd...)> fsimd, array< vect_t, sizeof...(ASimd)> &simd_in)`
- `bool getStatus () const`
- `string getTestName () const`
- `bool writeResultLine () const`
- `void writeDebugData () const`



### Static Public Attributes

- `static constexpr size_t vect_size` = `Simd::vect_size`

### Protected Attributes

- `size_t nb_lref`
- string `name`
- vector< `vectElt` > `inputs`
- vector< `vectElt` > `outputs_simd`
- vector< `vectElt` > `outputs_scalar`

## 12.328.1 Member Typedef Documentation

### 12.328.1.1 Element

```
template<typename Simd >
using Element = typename Simd::scalar_t
```

### 12.328.1.2 vect\_t

```
template<typename Simd >
using vect_t = typename Simd::vect_t
```

### 12.328.1.3 vectElt

```
template<typename Simd >
using vectElt = vector<Element>
```

### 12.328.1.4 enable\_if\_t

```
template<typename Simd >
template<bool B, typename T = void>
using enable_if_t = typename enable_if<B, T>::type
```

## 12.328.2 Constructor & Destructor Documentation

### 12.328.2.1 TestOneMethod()

```
template<typename Simd >
template<typename... AScal, typename RScal, typename... ASimd, typename RSimd, enable_if_t<
sizeof...(AScal)==sizeof...(ASimd)> * = nullptr, enable_if_t< count_nonconst_lvalue_reference<
AScal... >::n==count_nonconst_lvalue_reference< ASimd... >::n > * = nullptr, enable_if_t<
is_all_same< AScal... >::value > * = nullptr, enable_if_t< is_all_same< vect_t, ASimd...
>::value > * = nullptr>
TestOneMethod (
 function< RSimd(ASimd...)> fsimd,
 function< RScal(AScal...)> fscal,
 function< void(vector< vectElt > &)> genInputs,
 string fname) [inline]
```

## 12.328.3 Member Function Documentation

### 12.328.3.1 evaluate\_scalar\_method() [1/3]

```
template<typename Simd >
template<typename Ret, typename... AScal>
enable_if_t< is_all_same< Element, AScal... >::value &&std::is_convertible< Ret, Element >↔
::value, void > evaluate_scalar_method (
 function< Ret(AScal...)> fscal) [inline]
```

**12.328.3.2 evaluate\_scalar\_method() [2/3]**

```
template<typename Simd >
template<typename... AScal>
enable_if_t< is_all_same< vectElt, AScal... >::value, void > evaluate_scalar_method (
 function< vectElt(AScal...)> fscal) [inline]
```

**12.328.3.3 evaluate\_scalar\_method() [3/3]**

```
template<typename Simd >
template<typename... AScal>
enable_if_t< is_all_same< vectElt, AScal... >::value, void > evaluate_scalar_method (
 function< void(AScal...)> fscal) [inline]
```

**12.328.3.4 evaluate\_simd\_method() [1/2]**

```
template<typename Simd >
template<typename Ret , typename... ASimd>
enable_if_t< is_all_same< vect_t, ASimd... >::value &&std::is_convertible< Ret, vect_t >↔
::value, void > evaluate_simd_method (
 function< Ret(ASimd...)> fsimd,
 array< vect_t, sizeof...(ASimd)> & simd_in) [inline]
```

**12.328.3.5 evaluate\_simd\_method() [2/2]**

```
template<typename Simd >
template<typename... ASimd>
enable_if_t< is_all_same< vect_t, ASimd... >::value, void > evaluate_simd_method (
 function< void(ASimd...)> fsimd,
 array< vect_t, sizeof...(ASimd)> & simd_in) [inline]
```

**12.328.3.6 getStatus()**

```
template<typename Simd >
bool getStatus () const [inline]
```

**12.328.3.7 getTestName()**

```
template<typename Simd >
string getTestName () const [inline]
```

**12.328.3.8 writeResultLine()**

```
template<typename Simd >
bool writeResultLine () const [inline]
```

**12.328.3.9 writeDebugData()**

```
template<typename Simd >
void writeDebugData () const [inline]
```

**12.328.4 Field Documentation****12.328.4.1 vect\_size**

```
template<typename Simd >
constexpr size_t vect_size = Simd::vect_size [static], [constexpr]
```

**12.328.4.2 nb\_lref**

```
template<typename Simd >
size_t nb_lref [protected]
```

**12.328.4.3 name**

```
template<typename Simd >
string name [protected]
```

**12.328.4.4 inputs**

```
template<typename Simd >
vector<vectElt> inputs [protected]
```

**12.328.4.5 outputs\_simd**

```
template<typename Simd >
vector<vectElt> outputs_simd [protected]
```

**12.328.4.6 outputs\_scalar**

```
template<typename Simd >
vector<vectElt> outputs_scalar [protected]
```

The documentation for this class was generated from the following file:

- [test-simd.C](#)

**12.329 tfn\_minus Struct Reference**

```
#include <sparse_matrix_traits.h>
```

**Public Member Functions**

- `template<typename... Args>`  
`auto operator() (Args &&... args) const -> decltype(operator+(std::forward< Args >(args)...))`

**12.329.1 Member Function Documentation****12.329.1.1 operator>()()**

```
template<typename... Args>
auto operator() (
 Args &&... args) const -> decltype(operator+(std::forward<Args>(args)...))

[inline]
```

The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

**12.330 tfn\_minus\_eq Struct Reference**

```
#include <sparse_matrix_traits.h>
```

**Public Member Functions**

- `template<typename... Args>`  
`auto operator() (Args &&... args) const -> decltype(operator+(std::forward< Args >(args)...))`

## 12.330.1 Member Function Documentation

### 12.330.1.1 operator>()()

```
template<typename... Args>
auto operator() (
 Args &&... args) const -> decltype(operator+(std::forward<Args>(args)...))
[inline]
```

The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.331 tfn\_mul Struct Reference

```
#include <sparse_matrix_traits.h>
```

### Public Member Functions

- `template<typename... Args>`  
`auto operator() (Args &&... args) const -> decltype(operator+(std::forward< Args >(args)...))`

## 12.331.1 Member Function Documentation

### 12.331.1.1 operator>()()

```
template<typename... Args>
auto operator() (
 Args &&... args) const -> decltype(operator+(std::forward<Args>(args)...))
[inline]
```

The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.332 tfn\_mul\_eq Struct Reference

```
#include <sparse_matrix_traits.h>
```

### Public Member Functions

- `template<typename... Args>`  
`auto operator() (Args &&... args) const -> decltype(operator+(std::forward< Args >(args)...))`

## 12.332.1 Member Function Documentation

### 12.332.1.1 operator>()()

```
template<typename... Args>
auto operator() (
 Args &&... args) const -> decltype(operator+(std::forward<Args>(args)...))
[inline]
```

The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

## 12.333 tfn\_plus Struct Reference

```
#include <sparse_matrix_traits.h>
```

**Public Member Functions**

- `template<typename... Args>  
auto operator() (Args &&... args) const -> decltype(operator+(std::forward<Args>(args)...))`

**12.333.1 Member Function Documentation****12.333.1.1 operator>()()**

```
template<typename... Args>
auto operator() (
 Args &&... args) const -> decltype(operator+(std::forward<Args>(args)...))
[inline]
```

The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

**12.334 tfn\_plus\_eq Struct Reference**

```
#include <sparse_matrix_traits.h>
```

**Public Member Functions**

- `template<typename... Args>  
auto operator() (Args &&... args) const -> decltype(operator+(std::forward<Args>(args)...))`

**12.334.1 Member Function Documentation****12.334.1.1 operator>()()**

```
template<typename... Args>
auto operator() (
 Args &&... args) const -> decltype(operator+(std::forward<Args>(args)...))
[inline]
```

The documentation for this struct was generated from the following file:

- [sparse\\_matrix\\_traits.h](#)

**12.335 Threads Struct Reference**

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

**12.336 ThreeD Struct Reference**

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

**12.337 ThreeDAdaptive Struct Reference**

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

## 12.338 ThreeDInPlace Struct Reference

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

## 12.339 TRSMHelper< RecIterTrait, ParSeqTrait > Struct Template Reference

TRSM Helper.

### Public Member Functions

- `template<class Cut , class Param >  
TRSMHelper (ParSeqHelper::Parallel< Cut, Param > _PS)`
- `TRSMHelper (ParSeqHelper::Sequential _PS)`
- `template<typename RIT , typename PST >  
TRSMHelper (TRSMHelper< RIT, PST > & _TH)`
- `template<class Dom , class Algo = FFLAS::MMHelperAlgo::Winograd, class ModeT = typename FFLAS::ModeTraits<Dom>::value>  
FFLAS::MMHelper< Dom, Algo, ModeT, ParSeqTrait > pMMH (Dom &D, size_t m, size_t k, size_t n,  
ParSeqTrait p) const`
- `template<class Dom , class Algo = FFLAS::MMHelperAlgo::Winograd, class ModeT = typename FFLAS::ModeTraits<Dom>::value>  
FFLAS::MMHelper< Dom, Algo, ModeT, ParSeqTrait > pMMH (Dom &D, size_t m, size_t k, size_t n) const`

### Data Fields

- [ParSeqTrait parseq](#)

### 12.339.1 Detailed Description

```
template<typename RecIterTrait = StructureHelper::Recursive, typename ParSeqTrait = ParSeqHelper::Sequential>
struct FFLAS::TRSMHelper< RecIterTrait, ParSeqTrait >
```

TRSM Helper.

### 12.339.2 Constructor & Destructor Documentation

#### 12.339.2.1 TRSMHelper() [1/3]

```
template<typename RecIterTrait = StructureHelper::Recursive, typename ParSeqTrait = ParSeqHelper::Sequential>
template<class Cut , class Param >
TRSMHelper (
 ParSeqHelper::Parallel< Cut, Param > _PS) [inline]
```

#### 12.339.2.2 TRSMHelper() [2/3]

```
template<typename RecIterTrait = StructureHelper::Recursive, typename ParSeqTrait = ParSeqHelper::Sequential>
TRSMHelper (
 ParSeqHelper::Sequential _PS) [inline]
```

#### 12.339.2.3 TRSMHelper() [3/3]

```
template<typename RecIterTrait = StructureHelper::Recursive, typename ParSeqTrait = ParSeqHelper::Sequential>
template<typename RIT , typename PST >
TRSMHelper (
 TRSMHelper< RIT, PST > & _TH) [inline]
```

### 12.339.3 Member Function Documentation

#### 12.339.3.1 pMMH() [1/2]

```
template<typename RecIterTrait = StructureHelper::Recursive, typename ParSeqTrait = ParSeq↵
Helper::Sequential>
template<class Dom , class Algo = FFLAS::MMHelperAlgo::Winograd, class ModeT = typename FFLAS↵
::ModeTraits<Dom>::value>
FFLAS::MMHelper< Dom, Algo, ModeT, ParSeqTrait > pMMH (
 Dom & D,
 size_t m,
 size_t k,
 size_t n,
 ParSeqTrait p) const [inline]
```

#### 12.339.3.2 pMMH() [2/2]

```
template<typename RecIterTrait = StructureHelper::Recursive, typename ParSeqTrait = ParSeq↵
Helper::Sequential>
template<class Dom , class Algo = FFLAS::MMHelperAlgo::Winograd, class ModeT = typename FFLAS↵
::ModeTraits<Dom>::value>
FFLAS::MMHelper< Dom, Algo, ModeT, ParSeqTrait > pMMH (
 Dom & D,
 size_t m,
 size_t k,
 size_t n) const [inline]
```

### 12.339.4 Field Documentation

#### 12.339.4.1 parseq

```
template<typename RecIterTrait = StructureHelper::Recursive, typename ParSeqTrait = ParSeq↵
Helper::Sequential>
ParSeqTrait parseq
```

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

## 12.340 TwoD Struct Reference

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

## 12.341 TwoDAdaptive Struct Reference

The documentation for this struct was generated from the following file:

- [blockcuts.inl](#)

## 12.342 UnparametricTag Struct Reference

If the field uses a representation with infix operators.

```
#include <field-traits.h>
```

### 12.342.1 Detailed Description

If the field uses a representation with infix operators.

The documentation for this struct was generated from the following file:

- [field-traits.h](#)

## 12.343 `width< T >` Struct Template Reference

### Static Public Attributes

- `static constexpr size_t value = 2+2*sizeof(T)`

### 12.343.1 Field Documentation

#### 12.343.1.1 value

```
template<typename T >
constexpr size_t value = 2+2*sizeof(T) [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.344 `width< double >` Struct Reference

### Static Public Attributes

- `static constexpr size_t value = 24`

### 12.344.1 Field Documentation

#### 12.344.1.1 value

```
constexpr size_t value = 24 [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.345 `width< float >` Struct Reference

### Static Public Attributes

- `static constexpr size_t value = 16`

### 12.345.1 Field Documentation

#### 12.345.1.1 value

```
constexpr size_t value = 16 [static], [constexpr]
```

The documentation for this struct was generated from the following file:

- [test-simd.C](#)

## 12.346 Winograd Struct Reference

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)

## 12.347 WinogradPar Struct Reference

The documentation for this struct was generated from the following file:

- [fflas\\_helpers.inl](#)



# Chapter 13

## File Documentation

### 13.1 arithprog.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/utils/fflas_randommatrix.h"
#include <iostream>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include <ctime>
```

#### Macros

- #define `CUBE(x)`  $((x)*(x)*(x))$
- #define `GFOPS(m, n, r, t)`  $(2.7*CUBE(double(n)/1000.0))/t$

#### Typedefs

- typedef Givaro::Timer `TTimer`

#### Functions

- int `main` (int argc, char \*\*argv)

#### 13.1.1 Macro Definition Documentation

##### 13.1.1.1 CUBE

```
#define CUBE(
 x) ((x)*(x)*(x))
```

##### 13.1.1.2 GFOPS

```
#define GFOPS(
 m,
 n,
 r,
 t) (2.7*CUBE(double(n)/1000.0))/t
```

#### 13.1.2 Typedef Documentation

##### 13.1.2.1 TTimer

```
typedef Givaro::Timer TTimer
```

### 13.1.3 Function Documentation

#### 13.1.3.1 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.2 fsyrk.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include "fflas-ffpack/utils/fflas_randommatrix.h"
#include <ctime>
```

### Macros

- #define CUBE(x) ((x)\*(x)\*(x))
- #define GFOPS(n, t) (CUBE(double(n)/1000.0)/(3.0\*t))

### Typedefs

- typedef Givaro::Timer TTimer

### Functions

- int main ()

### 13.2.1 Macro Definition Documentation

#### 13.2.1.1 CUBE

```
#define CUBE(
 x) ((x)*(x)*(x))
```

#### 13.2.1.2 GFOPS

```
#define GFOPS(
 n,
 t) (CUBE(double(n)/1000.0)/(3.0*t))
```

### 13.2.2 Typedef Documentation

#### 13.2.2.1 TTimer

```
typedef Givaro::Timer TTimer
```

### 13.2.3 Function Documentation

#### 13.2.3.1 main()

```
int main (
 void)
```

## 13.3 fsytrf.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include "fflas-ffpack/utils/fflas_randommatrix.h"
#include <ctime>
```

### Macros

- `#define CUBE(x) ((x)*(x)*(x))`
- `#define GFOPS(n, t) (CUBE(double(n)/1000.0)/(3.0*t))`

### Typedefs

- `typedef Givaro::Timer TTimer`

### Functions

- `int main ()`

#### 13.3.1 Macro Definition Documentation

##### 13.3.1.1 CUBE

```
#define CUBE(
 x) ((x)*(x)*(x))
```

##### 13.3.1.2 GFOPS

```
#define GFOPS(
 n,
 t) (CUBE(double(n)/1000.0)/(3.0*t))
```

#### 13.3.2 Typedef Documentation

##### 13.3.2.1 TTimer

```
typedef Givaro::Timer TTimer
```

#### 13.3.3 Function Documentation

##### 13.3.3.1 main()

```
int main (
 void)
```

## 13.4 ftrtri.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/utils/fflas_randommatrix.h"
#include <iostream>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include <ctime>
```

**Macros**

- `#define CUBE(x) ((x)*(x)*(x))`
- `#define GFOPS(n, t) (CUBE(double(n)/1000.0)/(3.0*t))`

**Typedefs**

- `typedef Givaro::Timer TTimer`

**Functions**

- `int main ()`

**13.4.1 Macro Definition Documentation****13.4.1.1 CUBE**

```
#define CUBE(
 x) ((x)*(x)*(x))
```

**13.4.1.2 GFOPS**

```
#define GFOPS(
 n,
 t) (CUBE(double(n)/1000.0)/(3.0*t))
```

**13.4.2 Typedef Documentation****13.4.2.1 TTimer**

```
typedef Givaro::Timer TTimer
```

**13.4.3 Function Documentation****13.4.3.1 main()**

```
int main (
 void)
```

**13.5 winograd.C File Reference**

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/utils/fflas_randommatrix.h"
#include <iostream>
#include <fstream>
#include <givaro/modular.h>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include <ctime>
```

**Macros**

- `#define DOUBLE_TO_FLOAT_CROSSOVER 0`
- `#define GFOPS(n, t) (2.0/t*(double)n/1000.0*(double)n/1000.0*(double)n/1000.0)`

**Typedefs**

- `typedef Givaro::Timer TTimer`

## Functions

- template<class [Field](#) >  
bool [balanced](#) (const [Field](#) &)
- template<class T >  
bool [balanced](#) (const [Givaro::ModularBalanced](#)< T > &)
- int [main](#) ()

## 13.5.1 Macro Definition Documentation

### 13.5.1.1 DOUBLE\_TO\_FLOAT\_CROSSOVER

```
#define DOUBLE_TO_FLOAT_CROSSOVER 0
```

### 13.5.1.2 GFOPS

```
#define GFOPS(
 n,
 t) (2.0/t*(double)n/1000.0*(double)n/1000.0*(double)n/1000.0)
```

## 13.5.2 Typedef Documentation

### 13.5.2.1 TTimer

```
typedef Givaro::Timer TTimer
```

## 13.5.3 Function Documentation

### 13.5.3.1 [balanced\(\)](#) [1/2]

```
template<class Field >
bool balanced (
 const Field &)
```

### 13.5.3.2 [balanced\(\)](#) [2/2]

```
template<class T >
bool balanced (
 const Givaro::ModularBalanced< T > &)
```

### 13.5.3.3 [main\(\)](#)

```
int main (
 void)
```

## 13.6 benchmark-charpoly-mp.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/utils/Matio.h"
#include "fflas-ffpack/utils/args-parser.h"
```

## Macros

- #define [\\_\\_FFLASFFPACK\\_FORCE\\_SEQ](#)

## Functions

- int [main](#) (int argc, char \*\*argv)

### 13.6.1 Macro Definition Documentation

#### 13.6.1.1 \_\_FFLASFFPACK\_FORCE\_SEQ

```
#define __FFLASFFPACK_FORCE_SEQ
```

### 13.6.2 Function Documentation

#### 13.6.2.1 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.7 benchmark-charpoly.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular.h>
#include <givaro/givpoly1.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/utils/fflas_randommatrix.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/args-parser.h"
```

## Macros

- #define [\\_\\_FFLASFFPACK\\_OPENBLAS\\_NT\\_ALREADY\\_SET](#) 1

## Functions

- template<class [Field](#) >  
void [run\\_with\\_field](#) (int q, uint64\_t bits, size\_t n, size\_t d, size\_t iter, std::string file, int variant, uint64\_t seed)
- int [main](#) (int argc, char \*\*argv)

### 13.7.1 Macro Definition Documentation

#### 13.7.1.1 \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET

```
#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1
```

### 13.7.2 Function Documentation

#### 13.7.2.1 run\_with\_field()

```
template<class Field >
void run_with_field (
 int q,
 uint64_t bits,
 size_t n,
 size_t d,
 size_t iter,
 std::string file,
```

```
 int variant,
 uint64_t seed)
```

### 13.7.2.2 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.8 benchmark-checkers.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <stdlib.h>
#include <time.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/fflas_randommatrix.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/checkers/checkers_fflas.h"
#include "fflas-ffpack/checkers/checkers_ffpack.h"
#include <fstream>
```

### Macros

- `#define ENABLE_ALL_CHECKINGS 1`
- `#define _NR_TESTS 5`
- `#define _MAX_SIZE_MATRICES 1000`
- `#define CUBE(x) ((x)*(x)*(x))`

### Functions

- `int main (int argc, char **argv)`

## 13.8.1 Macro Definition Documentation

### 13.8.1.1 ENABLE\_ALL\_CHECKINGS

```
#define ENABLE_ALL_CHECKINGS 1
```

### 13.8.1.2 \_NR\_TESTS

```
#define _NR_TESTS 5
```

### 13.8.1.3 \_MAX\_SIZE\_MATRICES

```
#define _MAX_SIZE_MATRICES 1000
```

### 13.8.1.4 CUBE

```
#define CUBE(
 x) ((x)*(x)*(x))
```

## 13.8.2 Function Documentation

### 13.8.2.1 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.9 benchmark-dgemm.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular.h>
#include "fflas-ffpack/config-blas.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/args-parser.h"
```

### Macros

- #define [CBLAS\\_GEMM](#) cblas\_dgemm

### Typedefs

- typedef [FFLAS::Timer](#) TTimer
- typedef double [Floats](#)

### Functions

- int [main](#) (int argc, char \*\*argv)

## 13.9.1 Macro Definition Documentation

### 13.9.1.1 CBLAS\_GEMM

```
#define CBLAS_GEMM cblas_dgemm
```

## 13.9.2 Typedef Documentation

### 13.9.2.1 TTimer

```
typedef FFLAS::Timer TTimer
```

### 13.9.2.2 Floats

```
typedef double Floats
```

## 13.9.3 Function Documentation

### 13.9.3.1 main()

```
int main (
 int argc,
 char ** argv)
```



## 13.10 benchmark-dgetrf.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <vector>
#include <givaro/modular.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/args-parser.h"
```

### Macros

- `#define __FFLASFFPACK_HAVE_DGETRF 1`

### Typedefs

- `typedef FFLAS::Timer TTimer`

### Functions

- `int main (int argc, char **argv)`

## 13.10.1 Macro Definition Documentation

### 13.10.1.1 \_\_FFLASFFPACK\_HAVE\_DGETRF

```
#define __FFLASFFPACK_HAVE_DGETRF 1
```

## 13.10.2 Typedef Documentation

### 13.10.2.1 TTimer

```
typedef FFLAS::Timer TTimer
```

## 13.10.3 Function Documentation

### 13.10.3.1 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.11 benchmark-dgetri.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <vector>
#include <givaro/modular.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/args-parser.h"
```

### Typedefs

- `typedef FFLAS::Timer TTimer`

## Functions

- int [main](#) (int argc, char \*\*argv)

### 13.11.1 Typedef Documentation

#### 13.11.1.1 TTimer

```
typedef FFLAS::Timer TTimer
```

### 13.11.2 Function Documentation

#### 13.11.2.1 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.12 benchmark-dsytrf.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <vector>
#include <givaro/modular.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/args-parser.h"
```

## Macros

- #define [EFGGFF](#)(n, t, i) ( (double(n)/1000.\*double(n)/1000.\*double(n)/1000.0) / double(t) \* double(i) / 3.)

## Typedefs

- typedef [FFLAS::Timer](#) TTimer

## Functions

- int [main](#) (int argc, char \*\*argv)

### 13.12.1 Macro Definition Documentation

#### 13.12.1.1 EFGGFF

```
#define EFGGFF(
 n,
 t,
 i) ((double(n)/1000.*double(n)/1000.*double(n)/1000.0) / double(t) * double(i)
/ 3.)
```

### 13.12.2 Typedef Documentation

#### 13.12.2.1 TTimer

```
typedef FFLAS::Timer TTimer
```

### 13.12.3 Function Documentation

#### 13.12.3.1 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.13 benchmark-dtrsm.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/args-parser.h"
```

### Typedefs

- typedef [FFLAS::Timer](#) [TTimer](#)

### Functions

- int [main](#) (int argc, char \*\*argv)

### 13.13.1 Typedef Documentation

#### 13.13.1.1 TTimer

```
typedef FFLAS::Timer TTimer
```

### 13.13.2 Function Documentation

#### 13.13.2.1 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.14 benchmark-dtrtri.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/args-parser.h"
```

### Macros

- #define [\\_\\_FFLASFFPACK\\_HAVE\\_DTRTRI](#) 1

### Typedefs

- typedef [FFLAS::Timer](#) [TTimer](#)

## Functions

- int [main](#) (int argc, char \*\*argv)

### 13.14.1 Macro Definition Documentation

#### 13.14.1.1 \_\_FFLASFFPACK\_HAVE\_DTRTRI

```
#define __FFLASFFPACK_HAVE_DTRTRI 1
```

### 13.14.2 Typedef Documentation

#### 13.14.2.1 TTimer

```
typedef FFLAS::Timer TTimer
```

### 13.14.3 Function Documentation

#### 13.14.3.1 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.15 benchmark-fadd-lvl2.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/args-parser.h"
```

## Macros

- #define [\\_\\_FFLASFFPACK\\_OPENBLAS\\_NT\\_ALREADY\\_SET](#) 1

## Functions

- int [main](#) (int argc, char \*\*argv)

### 13.15.1 Macro Definition Documentation

#### 13.15.1.1 \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET

```
#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1
```

### 13.15.2 Function Documentation

#### 13.15.2.1 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.16 benchmark-fdot.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular.h>
#include <givaro/givrational.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/paladin/parallel.h"
#include "fflas-ffpack/paladin/fflas_plevel1.h"
#include "fflas-ffpack/utils/args-parser.h"
```

### Macros

- `#define \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET 1`

### Functions

- `template<class Field >`  
`Field::Element run\_with\_field (int q, size_t iter, size_t N, const uint64_t BS, const size_t p, const size_t threads, uint64_t seed)`
- `int main (int argc, char **argv)`

### 13.16.1 Macro Definition Documentation

#### 13.16.1.1 [\\_\\_FFLASFFPACK\\_OPENBLAS\\_NT\\_ALREADY\\_SET](#)

```
#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1
```

### 13.16.2 Function Documentation

#### 13.16.2.1 [run\\_with\\_field\(\)](#)

```
template<class Field >
Field::Element run_with_field (
 int q,
 size_t iter,
 size_t N,
 const uint64_t BS,
 const size_t p,
 const size_t threads,
 uint64_t seed)
```

#### 13.16.2.2 [main\(\)](#)

```
int main (
 int argc,
 char ** argv)
```

## 13.17 benchmark-fgemm-mp.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <typeinfo>
#include <vector>
#include <string>
```

```
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "givaro/modular-integer.h"
#include "givaro/givcaster.h"
#include "fflas-ffpack/paladin/parallel.h"
```

## Macros

- `#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1`
- `#define MG_DEFAULT MG_ACTIVE`
- `#define STD_RECINT_SIZE 8`

## Functions

- `template<typename Ints >`  
`int tmain ()`
- `int main (int argc, char **argv)`

## 13.17.1 Macro Definition Documentation

### 13.17.1.1 \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET

```
#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1
```

### 13.17.1.2 MG\_DEFAULT

```
#define MG_DEFAULT MG_ACTIVE
```

### 13.17.1.3 STD\_RECINT\_SIZE

```
#define STD_RECINT_SIZE 8
```

## 13.17.2 Function Documentation

### 13.17.2.1 tmain()

```
template<typename Ints >
int tmain ()
```

### 13.17.2.2 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.18 benchmark-fgemm-rns.C File Reference

```
#include "fflas-ffpack/fflas/fflas.h"
#include <iostream>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/args-parser.h"
```

## Macros

- `#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1`

## Typedefs

- typedef `FFPACK::rns_double` `RNS`
- typedef `FFPACK::RNSInteger< RNS >` `Field`
- typedef `Field::Element_ptr` `Element_ptr`
- typedef `Field::ConstElement_ptr` `ConstElement_ptr`
- typedef `StrategyParameter::Threads` `THREADS`
- typedef `StrategyParameter::Grain` `GRAIN`
- typedef `StrategyParameter::TwoD` `TWOD`
- typedef `StrategyParameter::TwoDAdaptive` `TWODA`
- typedef `StrategyParameter::ThreeD` `THREED`
- typedef `StrategyParameter::ThreeDAdaptive` `THREEDA`
- typedef `StrategyParameter::ThreeDInPlace` `THREEDIP`
- typedef `ParSeqHelper::Sequential` `PSeq`

## Functions

- int `main` (int argc, char \*argv[])

### 13.18.1 Macro Definition Documentation

#### 13.18.1.1 `__FFLASFFPACK_OPENBLAS_NT_ALREADY_SET`

```
#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1
```

### 13.18.2 Typedef Documentation

#### 13.18.2.1 `RNS`

```
typedef FFPACK::rns_double RNS
```

#### 13.18.2.2 `Field`

```
typedef FFPACK::RNSInteger<RNS> Field
```

#### 13.18.2.3 `Element_ptr`

```
typedef Field::Element_ptr Element_ptr
```

#### 13.18.2.4 `ConstElement_ptr`

```
typedef Field::ConstElement_ptr ConstElement_ptr
```

#### 13.18.2.5 `THREADS`

```
typedef StrategyParameter::Threads THREADS
```

#### 13.18.2.6 `GRAIN`

```
typedef StrategyParameter::Grain GRAIN
```

#### 13.18.2.7 `TWOD`

```
typedef StrategyParameter::TwoD TWOD
```

#### 13.18.2.8 `TWODA`

```
typedef StrategyParameter::TwoDAdaptive TWODA
```

**13.18.2.9 THREED**

```
typedef StrategyParameter::ThreeD THREED
```

**13.18.2.10 THREEDA**

```
typedef StrategyParameter::ThreeDAdaptive THREEDA
```

**13.18.2.11 THREEDIP**

```
typedef StrategyParameter::ThreeDInPlace THREEDIP
```

**13.18.2.12 PSeq**

```
typedef ParSeqHelper::Sequential PSeq
```

**13.18.3 Function Documentation****13.18.3.1 main()**

```
int main (
 int argc,
 char * argv[])
```

**13.19 benchmark-fgemm.C File Reference**

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/config-blas.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/args-parser.h"
```

**Macros**

- #define [CLASSIC\\_HYBRID](#)

**Functions**

- int [main](#) (int argc, char \*\*argv)

**13.19.1 Macro Definition Documentation****13.19.1.1 CLASSIC\_HYBRID**

```
#define CLASSIC_HYBRID
```

**13.19.2 Function Documentation****13.19.2.1 main()**

```
int main (
 int argc,
 char ** argv)
```



## 13.20 benchmark-fgemv-mp.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <typeinfo>
#include <vector>
#include <string>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "givaro/modular-integer.h"
#include "givaro/givcaster.h"
#include "fflas-ffpack/paladin/parallel.h"
```

### Macros

- `#define \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET 1`
- `#define MG\_DEFAULT MG_ACTIVE`
- `#define STD\_RECINT\_SIZE 8`

### Functions

- `template<typename T >`  
`std::ostream & write\_matrix (std::ostream &out, Givaro::Integer p, size_t m, size_t n, T *C, size_t ldc)`
- `template<typename Ints >`  
`int tmain ()`
- `int main (int argc, char **argv)`

## 13.20.1 Macro Definition Documentation

### 13.20.1.1 [\\_\\_FFLASFFPACK\\_OPENBLAS\\_NT\\_ALREADY\\_SET](#)

```
#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1
```

### 13.20.1.2 [MG\\_DEFAULT](#)

```
#define MG_DEFAULT MG_ACTIVE
```

### 13.20.1.3 [STD\\_RECINT\\_SIZE](#)

```
#define STD_RECINT_SIZE 8
```

## 13.20.2 Function Documentation

### 13.20.2.1 [write\\_matrix\(\)](#)

```
template<typename T >
std::ostream & write_matrix (
 std::ostream & out,
 Givaro::Integer p,
 size_t m,
 size_t n,
 T * C,
 size_t ldc)
```

### 13.20.2.2 [tmain\(\)](#)

```
template<typename Ints >
int tmain ()
```

### 13.20.2.3 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.21 benchmark-fgemv.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/config-blas.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "givaro/modular-integer.h"
#include "givaro/givcaster.h"
```

### Data Structures

- struct [need\\_field\\_characteristic](#)< Field >
- struct [need\\_field\\_characteristic](#)< Givaro::Modular< Field > >
- struct [need\\_field\\_characteristic](#)< Givaro::ModularBalanced< Field > >
- struct [compatible\\_data\\_type](#)< Field >
- struct [compatible\\_data\\_type](#)< Givaro::ZRing< float > >
- struct [compatible\\_data\\_type](#)< Givaro::ZRing< double > >

### Macros

- #define [\\_\\_FFLASFFPACK\\_OPENBLAS\\_NT\\_ALREADY\\_SET](#) 1

### Functions

- template<class [Field](#) , class RandIter , class Matrix , class Vector >  
void [fill\\_value](#) ([Field](#) &F, RandIter &Rand, Matrix &A, Vector &X, Vector &Y, size\_t m, size\_t k, size\_t incX, size\_t incY, size\_t lda, int NBK)
- template<class [Field](#) , class Matrix , class Vector >  
void [genData](#) ([Field](#) &F, Matrix &A, Vector &X, Vector &Y, size\_t m, size\_t k, size\_t incX, size\_t incY, size\_t lda, int NBK, uint64\_t bitsize, uint64\_t seed)
- template<class [Field](#) , class Matrix , class Vector >  
bool [check\\_result](#) ([Field](#) &F, size\_t m, size\_t lda, Matrix &A, Vector &X, size\_t incX, Vector &Y, size\_t incY)
- template<class [Field](#) , class Matrix , class Vector >  
bool [benchmark\\_with\\_timer](#) ([Field](#) &F, int p, Matrix &A, Vector &X, Vector &Y, size\_t m, size\_t k, size\_t incX, size\_t incY, size\_t lda, size\_t iters, int t, double &time, size\_t GrainSize)
- template<class [Field](#) , class arg >  
void [benchmark\\_disp](#) ([Field](#) &F, bool pass, double &time, size\_t iters, int p, size\_t m, size\_t k, arg &as)
- template<class [Field](#) , class arg >  
void [benchmark\\_in\\_Field](#) ([Field](#) &F, int p, size\_t m, size\_t k, int NBK, uint64\_t bitsize, uint64\_t seed, size\_t iters, int t, arg &as, size\_t GrainSize)
- template<class [Field](#) , class arg >  
void [benchmark\\_with\\_field](#) (int p, size\_t m, size\_t k, int NBK, uint64\_t bitsize, uint64\_t seed, size\_t iters, int t, arg &as, size\_t GrainSize)
- template<class [Field](#) , class arg >  
void [benchmark\\_with\\_field](#) (const Givaro::Integer &q, int p, size\_t m, size\_t k, int NBK, uint64\_t bitsize, uint64\_t seed, size\_t iters, int t, arg &as, size\_t GrainSize)
- int [main](#) (int argc, char \*\*argv)

## 13.21.1 Macro Definition Documentation

### 13.21.1.1 \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET

```
#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1
```

## 13.21.2 Function Documentation

### 13.21.2.1 fill\_value()

```
template<class Field , class RandIter , class Matrix , class Vector >
void fill_value (
 Field & F,
 RandIter & Rand,
 Matrix & A,
 Vector & X,
 Vector & Y,
 size_t m,
 size_t k,
 size_t incX,
 size_t incY,
 size_t lda,
 int NBK)
```

### 13.21.2.2 genData()

```
template<class Field , class Matrix , class Vector >
void genData (
 Field & F,
 Matrix & A,
 Vector & X,
 Vector & Y,
 size_t m,
 size_t k,
 size_t incX,
 size_t incY,
 size_t lda,
 int NBK,
 uint64_t bitsize,
 uint64_t seed)
```

### 13.21.2.3 check\_result()

```
template<class Field , class Matrix , class Vector >
bool check_result (
 Field & F,
 size_t m,
 size_t lda,
 Matrix & A,
 Vector & X,
 size_t incX,
 Vector & Y,
 size_t incY)
```

### 13.21.2.4 benchmark\_with\_timer()

```
template<class Field , class Matrix , class Vector >
bool benchmark_with_timer (
 Field & F,
 int p,
```

```

Matrix & A,
Vector & X,
Vector & Y,
size_t m,
size_t k,
size_t incX,
size_t incY,
size_t lda,
size_t iters,
int t,
double & time,
size_t GrainSize)

```

#### 13.21.2.5 benchmark\_disp()

```

template<class Field , class arg >
void benchmark_disp (
 Field & F,
 bool pass,
 double & time,
 size_t iters,
 int p,
 size_t m,
 size_t k,
 arg & as)

```

#### 13.21.2.6 benchmark\_in\_Field()

```

template<class Field , class arg >
void benchmark_in_Field (
 Field & F,
 int p,
 size_t m,
 size_t k,
 int NBK,
 uint64_t bitsize,
 uint64_t seed,
 size_t iters,
 int t,
 arg & as,
 size_t GrainSize)

```

#### 13.21.2.7 benchmark\_with\_field() [1/2]

```

template<class Field , class arg >
void benchmark_with_field (
 int p,
 size_t m,
 size_t k,
 int NBK,
 uint64_t bitsize,
 uint64_t seed,
 size_t iters,
 int t,
 arg & as,
 size_t GrainSize)

```

**13.21.2.8 benchmark\_with\_field() [2/2]**

```
template<class Field , class arg >
void benchmark_with_field (
 const Givaro::Integer & q,
 int p,
 size_t m,
 size_t k,
 int NBK,
 uint64_t bitsize,
 uint64_t seed,
 size_t iters,
 int t,
 arg & as,
 size_t GrainSize)
```

**13.21.2.9 main()**

```
int main (
 int argc,
 char ** argv)
```

**13.22 benchmark-fgesv.C File Reference**

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/args-parser.h"
```

**Macros**

- #define `__FFLASFFPACK_OPENBLAS_NT_ALREADY_SET` 1

**Functions**

- int `main` (int argc, char \*\*argv)

**13.22.1 Macro Definition Documentation****13.22.1.1 \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET**

```
#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1
```

**13.22.2 Function Documentation****13.22.2.1 main()**

```
int main (
 int argc,
 char ** argv)
```

**13.23 benchmark-fsyr2k.C File Reference**

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
```

```
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/config-blas.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/args-parser.h"
```

## Functions

- int [main](#) (int argc, char \*\*argv)

### 13.23.1 Function Documentation

#### 13.23.1.1 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.24 benchmark-fsyrrk.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/args-parser.h"
```

## Macros

- #define [\\_\\_FFLASFFPACK\\_OPENBLAS\\_NT\\_ALREADY\\_SET](#) 1

## Functions

- int [main](#) (int argc, char \*\*argv)

### 13.24.1 Macro Definition Documentation

#### 13.24.1.1 \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET

```
#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1
```

### 13.24.2 Function Documentation

#### 13.24.2.1 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.25 benchmark-fsytrf.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/timer.h"
```

```
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/args-parser.h"
```

## Macros

- `#define __FFPACK_FSYTRF_BC_CROUT`
- `#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1`
- `#define CUBE(x) ((x)*(x)*(x))`

## Functions

- `int main (int argc, char **argv)`

### 13.25.1 Macro Definition Documentation

#### 13.25.1.1 \_\_FFPACK\_FSYTRF\_BC\_CROUT

```
#define __FFPACK_FSYTRF_BC_CROUT
```

#### 13.25.1.2 \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET

```
#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1
```

#### 13.25.1.3 CUBE

```
#define CUBE(
 x) ((x)*(x)*(x))
```

### 13.25.2 Function Documentation

#### 13.25.2.1 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.26 benchmark-fftrsm-mp.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <vector>
#include <string>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "givaro/modular-integer.h"
```

## Macros

- `#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1`

## Functions

- `int main (int argc, char **argv)`

### 13.26.1 Macro Definition Documentation

#### 13.26.1.1 \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET

```
#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1
```

### 13.26.2 Function Documentation

#### 13.26.2.1 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.27 benchmark-ftsm.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/args-parser.h"
```

#### Macros

- [#define \\_\\_FFLASFFPACK\\_OPENBLAS\\_NT\\_ALREADY\\_SET 1](#)

#### Functions

- [int main](#) (int argc, char \*\*argv)

### 13.27.1 Macro Definition Documentation

#### 13.27.1.1 \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET

```
#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1
```

### 13.27.2 Function Documentation

#### 13.27.2.1 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.28 benchmark-ftsv.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/args-parser.h"
```

#### Macros

- [#define \\_\\_FFLASFFPACK\\_OPENBLAS\\_NT\\_ALREADY\\_SET 1](#)



## Functions

- int [main](#) (int argc, char \*\*argv)

### 13.28.1 Macro Definition Documentation

#### 13.28.1.1 \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET

```
#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1
```

### 13.28.2 Function Documentation

#### 13.28.2.1 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.29 benchmark-fftrtri.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/args-parser.h"
```

## Macros

- #define [\\_\\_FFLASFFPACK\\_OPENBLAS\\_NT\\_ALREADY\\_SET](#) 1
- #define [CUBE](#)(x) ((x)\*(x)\*(x))

## Functions

- int [main](#) (int argc, char \*\*argv)

### 13.29.1 Macro Definition Documentation

#### 13.29.1.1 \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET

```
#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1
```

#### 13.29.1.2 CUBE

```
#define CUBE(
 x) ((x)*(x)*(x))
```

### 13.29.2 Function Documentation

#### 13.29.2.1 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.30 benchmark-inverse.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/args-parser.h"
```

### Macros

- `#define CUBE(x) ((x)*(x)*(x))`

### Functions

- `int main (int argc, char **argv)`

### 13.30.1 Macro Definition Documentation

#### 13.30.1.1 CUBE

```
#define CUBE(
 x) ((x)*(x)*(x))
```

### 13.30.2 Function Documentation

#### 13.30.2.1 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.31 benchmark-lqmp.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <vector>
#include <string>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "givaro/modular-integer.h"
```

### Functions

- `int main (int argc, char **argv)`

### 13.31.1 Function Documentation

#### 13.31.1.1 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.32 benchmark-lqup.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/args-parser.h"
```

### Macros

- `#define CUBE(x) ((x)*(x)*(x))`

### Functions

- `int main (int argc, char **argv)`

## 13.32.1 Macro Definition Documentation

### 13.32.1.1 CUBE

```
#define CUBE(
 x) ((x)*(x)*(x))
```

## 13.32.2 Function Documentation

### 13.32.2.1 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.33 benchmark-pluq.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <givaro/modular.h>
#include <givaro/givranditer.h>
#include <iostream>
#include "fflas-ffpack/config-blas.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/fflas_randommatrix.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/ffpack/ffpack.h"
```

### Macros

- `#define _FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1`
- `#define CUBE(x) ((x)*(x)*(x))`

### Typedefs

- `typedef Givaro::ModularBalanced< double > Field`

## Functions

- void `verification_PLUQ` (const `Field` &`F`, typename `Field::Element` \*`B`, typename `Field::Element` \*`A`, `size_t` \*`P`, `size_t` \*`Q`, `size_t` `m`, `size_t` `n`, `size_t` `R`)
- void `Rec_Initialize` (`Field` &`F`, `Field::Element` \*`C`, `size_t` `m`, `size_t` `n`, `size_t` `ldc`)
- int `main` (int `argc`, char \*\*`argv`)

## 13.33.1 Macro Definition Documentation

### 13.33.1.1 \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET

```
#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1
```

### 13.33.1.2 CUBE

```
#define CUBE(
 x) ((x)*(x)*(x))
```

## 13.33.2 Typedef Documentation

### 13.33.2.1 Field

```
typedef Givaro::ModularBalanced<double> Field
```

## 13.33.3 Function Documentation

### 13.33.3.1 verification\_PLUQ()

```
void verification_PLUQ (
 const Field & F,
 typename Field::Element * B,
 typename Field::Element * A,
 size_t * P,
 size_t * Q,
 size_t m,
 size_t n,
 size_t R)
```

### 13.33.3.2 Rec\_Initialize()

```
void Rec_Initialize (
 Field & F,
 Field::Element * C,
 size_t m,
 size_t n,
 size_t ldc)
```

### 13.33.3.3 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.34 benchmark-quasisep.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular.h>
#include <givaro/givpoly1.h>
```

```
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/utils/fflas_randommatrix.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/args-parser.h"
```

## Macros

- `#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1`

## Functions

- `template<class Field >`  
`void run_with_field (int q, size_t n, size_t m, size_t t, size_t r, size_t iter, uint64_t seed)`
- `int main (int argc, char **argv)`

### 13.34.1 Macro Definition Documentation

#### 13.34.1.1 \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET

```
#define __FFLASFFPACK_OPENBLAS_NT_ALREADY_SET 1
```

### 13.34.2 Function Documentation

#### 13.34.2.1 run\_with\_field()

```
template<class Field >
void run_with_field (
 int q,
 size_t n,
 size_t m,
 size_t t,
 size_t r,
 size_t iter,
 uint64_t seed)
```

#### 13.34.2.2 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.35 benchmark-storage-transpose.C File Reference

```
#include <iomanip>
#include <iostream>
#include <random>
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/fflas_memory.h"
#include <givaro/modular.h>
#include <recint/rint.h>
#include "fflas-ffpack/fflas/fflas_transpose.h"
```

## Data Structures

- class [Bench<Elt>](#)

## Functions

- int [main](#) (int argc, char \*\*argv)

## 13.35.1 Function Documentation

### 13.35.1.1 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.36 benchmark-wino.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <fstream>
#include <givaro/modular.h>
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/utils/args-parser.h"
```

## Macros

- #define [CUBE](#)(x) ((x)\*(x)\*(x))

## Functions

- template<class [Field](#) >  
void [launch\\_wino](#) (const [Field](#) &F, const size\_t &n, const size\_t &NB, const size\_t &wino, const bool &asmax, const size\_t &seed, const bool compare)
- int [main](#) (int argc, char \*\*argv)

## 13.36.1 Macro Definition Documentation

### 13.36.1.1 CUBE

```
#define CUBE(
 x) ((x)*(x)*(x))
```

## 13.36.2 Function Documentation

### 13.36.2.1 launch\_wino()

```
template<class Field >
void launch_wino (
 const Field & F,
 const size_t & n,
 const size_t & NB,
 const size_t & wino,
 const bool & asmax,
 const size_t & seed,
 const bool compare)
```

### 13.36.2.2 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.37 config.h File Reference

### Macros

- #define [HAVE\\_BIG\\_ENDIAN](#) 1
- #define [HAVE\\_BLAS](#) 1
- #define [HAVE\\_CBLAS](#) 1
- #define [HAVE\\_CXX11](#) 1
- #define [HAVE\\_DLFCN\\_H](#) 1
- #define [HAVE\\_FLOAT\\_H](#) 1
- #define [HAVE\\_INT128](#) 1
- #define [HAVE\\_INTTYPES\\_H](#) 1
- #define [HAVE\\_LAPACK](#) 1
- #define [HAVE\\_LIMITS\\_H](#) 1
- #define [HAVE\\_PTHREAD\\_H](#) 1
- #define [HAVE\\_STDDEF\\_H](#) 1
- #define [HAVE\\_STDINT\\_H](#) 1
- #define [HAVE\\_STDIO\\_H](#) 1
- #define [HAVE\\_STDLIB\\_H](#) 1
- #define [HAVE\\_STRINGS\\_H](#) 1
- #define [HAVE\\_STRING\\_H](#) 1
- #define [HAVE\\_SYS\\_STAT\\_H](#) 1
- #define [HAVE\\_SYS\\_TIME\\_H](#) 1
- #define [HAVE\\_SYS\\_TYPES\\_H](#) 1
- #define [HAVE\\_UNISTD\\_H](#) 1
- #define [LT\\_OBJDIR](#) ".libs/"
- #define [OPENBLAS\\_NUM\\_THREADS](#) 1
- #define [PACKAGE](#) "fflas-ffpack"
- #define [PACKAGE\\_BUGREPORT](#) "ffpack-devel@googlegroups.com"
- #define [PACKAGE\\_NAME](#) "FFLAS-FFPACK"
- #define [PACKAGE\\_STRING](#) "FFLAS-FFPACK 2.5.0"
- #define [PACKAGE\\_TARNAME](#) "fflas-ffpack"
- #define [PACKAGE\\_URL](#) "https://github.com/linbox-team/fflas-ffpack"
- #define [PACKAGE\\_VERSION](#) "2.5.0"
- #define [SIZEOF\\_CHAR](#) 1
- #define [SIZEOF\\_INT](#) 4
- #define [SIZEOF\\_LONG](#) 8
- #define [SIZEOF\\_LONG\\_LONG](#) 8
- #define [SIZEOF\\_SHORT](#) 2
- #define [SIZEOF\\_\\_\\_INT64\\_T](#) 8
- #define [STDC\\_HEADERS](#) 1
- #define [USE\\_OPENMP](#) 1
- #define [VERSION](#) "2.5.0"

### 13.37.1 Macro Definition Documentation

#### 13.37.1.1 HAVE\_BIG\_ENDIAN

```
#define HAVE_BIG_ENDIAN 1
```

**13.37.1.2 HAVE\_BLAS**

```
#define HAVE_BLAS 1
```

**13.37.1.3 HAVE\_CBLAS**

```
#define HAVE_CBLAS 1
```

**13.37.1.4 HAVE\_CXX11**

```
#define HAVE_CXX11 1
```

**13.37.1.5 HAVE\_DLFCN\_H**

```
#define HAVE_DLFCN_H 1
```

**13.37.1.6 HAVE\_FLOAT\_H**

```
#define HAVE_FLOAT_H 1
```

**13.37.1.7 HAVE\_INT128**

```
#define HAVE_INT128 1
```

**13.37.1.8 HAVE\_INTPYPES\_H**

```
#define HAVE_INTPYPES_H 1
```

**13.37.1.9 HAVE\_LAPACK**

```
#define HAVE_LAPACK 1
```

**13.37.1.10 HAVE\_LIMITS\_H**

```
#define HAVE_LIMITS_H 1
```

**13.37.1.11 HAVE\_PTHREAD\_H**

```
#define HAVE_PTHREAD_H 1
```

**13.37.1.12 HAVE\_STDDEF\_H**

```
#define HAVE_STDDEF_H 1
```

**13.37.1.13 HAVE\_STDINT\_H**

```
#define HAVE_STDINT_H 1
```

**13.37.1.14 HAVE\_STDIO\_H**

```
#define HAVE_STDIO_H 1
```

**13.37.1.15 HAVE\_STDLIB\_H**

```
#define HAVE_STDLIB_H 1
```

**13.37.1.16 HAVE\_STRINGS\_H**

```
#define HAVE_STRINGS_H 1
```



**13.37.1.17 HAVE\_STRING\_H**

```
#define HAVE_STRING_H 1
```

**13.37.1.18 HAVE\_SYS\_STAT\_H**

```
#define HAVE_SYS_STAT_H 1
```

**13.37.1.19 HAVE\_SYS\_TIME\_H**

```
#define HAVE_SYS_TIME_H 1
```

**13.37.1.20 HAVE\_SYS\_TYPES\_H**

```
#define HAVE_SYS_TYPES_H 1
```

**13.37.1.21 HAVE\_UNISTD\_H**

```
#define HAVE_UNISTD_H 1
```

**13.37.1.22 LT\_OBJDIR**

```
#define LT_OBJDIR ".libs/"
```

**13.37.1.23 OPENBLAS\_NUM\_THREADS**

```
#define OPENBLAS_NUM_THREADS 1
```

**13.37.1.24 PACKAGE**

```
#define PACKAGE "fflas-ffpack"
```

**13.37.1.25 PACKAGE\_BUGREPORT**

```
#define PACKAGE_BUGREPORT "ffpack-devel@googlegroups.com"
```

**13.37.1.26 PACKAGE\_NAME**

```
#define PACKAGE_NAME "FFLAS-FFPACK"
```

**13.37.1.27 PACKAGE\_STRING**

```
#define PACKAGE_STRING "FFLAS-FFPACK 2.5.0"
```

**13.37.1.28 PACKAGE\_TARNAME**

```
#define PACKAGE_TARNAME "fflas-ffpack"
```

**13.37.1.29 PACKAGE\_URL**

```
#define PACKAGE_URL "https://github.com/linbox-team/fflas-ffpack"
```

**13.37.1.30 PACKAGE\_VERSION**

```
#define PACKAGE_VERSION "2.5.0"
```

**13.37.1.31 SIZEOF\_CHAR**

```
#define SIZEOF_CHAR 1
```

**13.37.1.32    `SIZEOF_INT`**

```
#define SIZEOF_INT 4
```

**13.37.1.33    `SIZEOF_LONG`**

```
#define SIZEOF_LONG 8
```

**13.37.1.34    `SIZEOF_LONG_LONG`**

```
#define SIZEOF_LONG_LONG 8
```

**13.37.1.35    `SIZEOF_SHORT`**

```
#define SIZEOF_SHORT 2
```

**13.37.1.36    `SIZEOF__INT64_T`**

```
#define SIZEOF__INT64_T 8
```

**13.37.1.37    `STDC_HEADERS`**

```
#define STDC_HEADERS 1
```

**13.37.1.38    `USE_OPENMP`**

```
#define USE_OPENMP 1
```

**13.37.1.39    `VERSION`**

```
#define VERSION "2.5.0"
```

**13.38    `fflas-ffpack/config.h` File Reference****Macros**

- [#define `\_\_FFLASFFPACK\_HAVE\_BIG\_ENDIAN` 1](#)
- [#define `\_\_FFLASFFPACK\_HAVE\_BLAS` 1](#)
- [#define `\_\_FFLASFFPACK\_HAVE\_CBLAS` 1](#)
- [#define `\_\_FFLASFFPACK\_HAVE\_CXX11` 1](#)
- [#define `\_\_FFLASFFPACK\_HAVE\_DLFCN\_H` 1](#)
- [#define `\_\_FFLASFFPACK\_HAVE\_FLOAT\_H` 1](#)
- [#define `\_\_FFLASFFPACK\_HAVE\_INT128` 1](#)
- [#define `\_\_FFLASFFPACK\_HAVE\_INTPTR\_T` 1](#)
- [#define `\_\_FFLASFFPACK\_HAVE\_LAPACK` 1](#)
- [#define `\_\_FFLASFFPACK\_HAVE\_LIMITS\_H` 1](#)
- [#define `\_\_FFLASFFPACK\_HAVE\_PTHREAD\_H` 1](#)
- [#define `\_\_FFLASFFPACK\_HAVE\_STDDEF\_H` 1](#)
- [#define `\_\_FFLASFFPACK\_HAVE\_STDINT\_H` 1](#)
- [#define `\_\_FFLASFFPACK\_HAVE\_STDIO\_H` 1](#)
- [#define `\_\_FFLASFFPACK\_HAVE\_STDLIB\_H` 1](#)
- [#define `\_\_FFLASFFPACK\_HAVE\_STRINGS\_H` 1](#)
- [#define `\_\_FFLASFFPACK\_HAVE\_STRING\_H` 1](#)
- [#define `\_\_FFLASFFPACK\_HAVE\_SYS\_STAT\_H` 1](#)
- [#define `\_\_FFLASFFPACK\_HAVE\_SYS\_TIME\_H` 1](#)
- [#define `\_\_FFLASFFPACK\_HAVE\_SYS\_TYPES\_H` 1](#)
- [#define `\_\_FFLASFFPACK\_HAVE\_UNISTD\_H` 1](#)
- [#define `\_\_FFLASFFPACK\_LT\_OBJDIR` ".libs/"](#)

- `#define __FFLASFFPACK_OPENBLAS_NUM_THREADS 1`
- `#define __FFLASFFPACK_PACKAGE "fflas-ffpack"`
- `#define __FFLASFFPACK_PACKAGE_BUGREPORT "ffpack-devel@googlegroups.com"`
- `#define __FFLASFFPACK_PACKAGE_NAME "FFLAS-FFPACK"`
- `#define __FFLASFFPACK_PACKAGE_STRING "FFLAS-FFPACK 2.5.0"`
- `#define __FFLASFFPACK_PACKAGE_TARNAME "fflas-ffpack"`
- `#define __FFLASFFPACK_PACKAGE_URL "https://github.com/linbox-team/fflas-ffpack"`
- `#define __FFLASFFPACK_PACKAGE_VERSION "2.5.0"`
- `#define __FFLASFFPACK_SIZEOF_CHAR 1`
- `#define __FFLASFFPACK_SIZEOF_INT 4`
- `#define __FFLASFFPACK_SIZEOF_LONG 8`
- `#define __FFLASFFPACK_SIZEOF_LONG_LONG 8`
- `#define __FFLASFFPACK_SIZEOF_SHORT 2`
- `#define __FFLASFFPACK_SIZEOF__INT64_T 8`
- `#define __FFLASFFPACK_STDC_HEADERS 1`
- `#define __FFLASFFPACK_USE_OPENMP 1`
- `#define __FFLASFFPACK_VERSION "2.5.0"`

### 13.38.1 Macro Definition Documentation

#### 13.38.1.1 `__FFLASFFPACK_HAVE_BIG_ENDIAN`

```
#define __FFLASFFPACK_HAVE_BIG_ENDIAN 1
```

#### 13.38.1.2 `__FFLASFFPACK_HAVE_BLAS`

```
#define __FFLASFFPACK_HAVE_BLAS 1
```

#### 13.38.1.3 `__FFLASFFPACK_HAVE_CBLAS`

```
#define __FFLASFFPACK_HAVE_CBLAS 1
```

#### 13.38.1.4 `__FFLASFFPACK_HAVE_CXX11`

```
#define __FFLASFFPACK_HAVE_CXX11 1
```

#### 13.38.1.5 `__FFLASFFPACK_HAVE_DLFCN_H`

```
#define __FFLASFFPACK_HAVE_DLFCN_H 1
```

#### 13.38.1.6 `__FFLASFFPACK_HAVE_FLOAT_H`

```
#define __FFLASFFPACK_HAVE_FLOAT_H 1
```

#### 13.38.1.7 `__FFLASFFPACK_HAVE_INT128`

```
#define __FFLASFFPACK_HAVE_INT128 1
```

#### 13.38.1.8 `__FFLASFFPACK_HAVE_INTTYPES_H`

```
#define __FFLASFFPACK_HAVE_INTTYPES_H 1
```

#### 13.38.1.9 `__FFLASFFPACK_HAVE_LAPACK`

```
#define __FFLASFFPACK_HAVE_LAPACK 1
```

#### 13.38.1.10 `__FFLASFFPACK_HAVE_LIMITS_H`

```
#define __FFLASFFPACK_HAVE_LIMITS_H 1
```

**13.38.1.11 \_\_FFLASFFPACK\_HAVE\_PTHREAD\_H**

```
#define __FFLASFFPACK_HAVE_PTHREAD_H 1
```

**13.38.1.12 \_\_FFLASFFPACK\_HAVE\_STDDEF\_H**

```
#define __FFLASFFPACK_HAVE_STDDEF_H 1
```

**13.38.1.13 \_\_FFLASFFPACK\_HAVE\_STDINT\_H**

```
#define __FFLASFFPACK_HAVE_STDINT_H 1
```

**13.38.1.14 \_\_FFLASFFPACK\_HAVE\_STDIO\_H**

```
#define __FFLASFFPACK_HAVE_STDIO_H 1
```

**13.38.1.15 \_\_FFLASFFPACK\_HAVE\_STDLIB\_H**

```
#define __FFLASFFPACK_HAVE_STDLIB_H 1
```

**13.38.1.16 \_\_FFLASFFPACK\_HAVE\_STRINGS\_H**

```
#define __FFLASFFPACK_HAVE_STRINGS_H 1
```

**13.38.1.17 \_\_FFLASFFPACK\_HAVE\_STRING\_H**

```
#define __FFLASFFPACK_HAVE_STRING_H 1
```

**13.38.1.18 \_\_FFLASFFPACK\_HAVE\_SYS\_STAT\_H**

```
#define __FFLASFFPACK_HAVE_SYS_STAT_H 1
```

**13.38.1.19 \_\_FFLASFFPACK\_HAVE\_SYS\_TIME\_H**

```
#define __FFLASFFPACK_HAVE_SYS_TIME_H 1
```

**13.38.1.20 \_\_FFLASFFPACK\_HAVE\_SYS\_TYPES\_H**

```
#define __FFLASFFPACK_HAVE_SYS_TYPES_H 1
```

**13.38.1.21 \_\_FFLASFFPACK\_HAVE\_UNISTD\_H**

```
#define __FFLASFFPACK_HAVE_UNISTD_H 1
```

**13.38.1.22 \_\_FFLASFFPACK\_LT\_OBJDIR**

```
#define __FFLASFFPACK_LT_OBJDIR ".libs/"
```

**13.38.1.23 \_\_FFLASFFPACK\_OPENBLAS\_NUM\_THREADS**

```
#define __FFLASFFPACK_OPENBLAS_NUM_THREADS 1
```

**13.38.1.24 \_\_FFLASFFPACK\_PACKAGE**

```
#define __FFLASFFPACK_PACKAGE "fflas-ffpack"
```

**13.38.1.25 \_\_FFLASFFPACK\_PACKAGE\_BUGREPORT**

```
#define __FFLASFFPACK_PACKAGE_BUGREPORT "ffpack-devel@googlegroups.com"
```

**13.38.1.26 \_\_FFLASFFPACK\_PACKAGE\_NAME**

```
#define __FFLASFFPACK_PACKAGE_NAME "FFLAS-FFPACK"
```

**13.38.1.27 \_\_FFLASFFPACK\_PACKAGE\_STRING**

```
#define __FFLASFFPACK_PACKAGE_STRING "FFLAS-FFPACK 2.5.0"
```

**13.38.1.28 \_\_FFLASFFPACK\_PACKAGE\_TARNAME**

```
#define __FFLASFFPACK_PACKAGE_TARNAME "fflas-ffpack"
```

**13.38.1.29 \_\_FFLASFFPACK\_PACKAGE\_URL**

```
#define __FFLASFFPACK_PACKAGE_URL "https://github.com/linbox-team/fflas-ffpack"
```

**13.38.1.30 \_\_FFLASFFPACK\_PACKAGE\_VERSION**

```
#define __FFLASFFPACK_PACKAGE_VERSION "2.5.0"
```

**13.38.1.31 \_\_FFLASFFPACK\_SIZEOF\_CHAR**

```
#define __FFLASFFPACK_SIZEOF_CHAR 1
```

**13.38.1.32 \_\_FFLASFFPACK\_SIZEOF\_INT**

```
#define __FFLASFFPACK_SIZEOF_INT 4
```

**13.38.1.33 \_\_FFLASFFPACK\_SIZEOF\_LONG**

```
#define __FFLASFFPACK_SIZEOF_LONG 8
```

**13.38.1.34 \_\_FFLASFFPACK\_SIZEOF\_LONG\_LONG**

```
#define __FFLASFFPACK_SIZEOF_LONG_LONG 8
```

**13.38.1.35 \_\_FFLASFFPACK\_SIZEOF\_SHORT**

```
#define __FFLASFFPACK_SIZEOF_SHORT 2
```

**13.38.1.36 \_\_FFLASFFPACK\_SIZEOF\_\_INT64\_T**

```
#define __FFLASFFPACK_SIZEOF__INT64_T 8
```

**13.38.1.37 \_\_FFLASFFPACK\_STDC\_HEADERS**

```
#define __FFLASFFPACK_STDC_HEADERS 1
```

**13.38.1.38 \_\_FFLASFFPACK\_USE\_OPENMP**

```
#define __FFLASFFPACK_USE_OPENMP 1
```

**13.38.1.39 \_\_FFLASFFPACK\_VERSION**

```
#define __FFLASFFPACK_VERSION "2.5.0"
```

## 13.39 mainpage.dox File Reference

## 13.40 autotune/charpoly.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/utils/fflas_randommatrix.h"
#include <iostream>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include <ctime>
```

### Macros

- `#define CUBE(x) ((x)*(x)*(x))`
- `#define GFOPS(m, n, r, t) (2.7*CUBE(double(n)/1000.0))/t`

### Typedefs

- `typedef Givaro::Timer TTimer`

### Functions

- `int main ()`

## 13.40.1 Macro Definition Documentation

### 13.40.1.1 CUBE

```
#define CUBE(
 x) ((x) * (x) * (x))
```

### 13.40.1.2 GFOPS

```
#define GFOPS(
 m,
 n,
 r,
 t) (2.7*CUBE(double(n)/1000.0)) / t
```

## 13.40.2 Typedef Documentation

### 13.40.2.1 TTimer

```
typedef Givaro::Timer TTimer
```

## 13.40.3 Function Documentation

### 13.40.3.1 main()

```
int main (
 void)
```

## 13.41 examples/charpoly.C File Reference

```
#include <iostream>
#include "fflas-ffpack/fflas-ffpack.h"
```

**Functions**

- int `main` (int argc, char \*\*argv)

*This example computes the characteristic polynomial of a matrix over a defined finite field.*

**13.41.1 Function Documentation****13.41.1.1 main()**

```
int main (
 int argc,
 char ** argv)
```

This example computes the characteristic polynomial of a matrix over a defined finite field.  
Outputs the characteristic polynomial.

**13.42 det.C File Reference**

```
#include <givaro/modular.h>
#include <iostream>
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/fflas_io.h"
```

**Functions**

- int `main` (int argc, char \*\*argv)

*This example computes the determinant of a matrix over a defined finite field.*

**13.42.1 Function Documentation****13.42.1.1 main()**

```
int main (
 int argc,
 char ** argv)
```

This example computes the determinant of a matrix over a defined finite field.  
Outputs the determinant.

**13.43 matmul.C File Reference**

```
#include <iostream>
#include "fflas-ffpack/fflas-ffpack.h"
```

**Functions**

- int `main` (int argc, char \*\*argv)

*This example computes the matrix multiplication over a defined finite field.*

**13.43.1 Function Documentation****13.43.1.1 main()**

```
int main (
 int argc,
 char ** argv)
```

This example computes the matrix multiplication over a defined finite field.  
Outputs the product of the matrix given as input.

## 13.44 autotune/pluq.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/utils/fflas_randommatrix.h"
#include <iostream>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include <ctime>
```

### Macros

- `#define CUBE(x) ((x)*(x)*(x))`
- `#define GFOPS(m, n, r, t) (2.0/3.0*CUBE(double(n)/1000.0) +2*m/1000.0*n/1000.0*double(r)/1000.0 - double(r)/1000.0*double(r)/1000.0*(m+n)/1000)/t`

### Typedefs

- `typedef Givaro::Timer TTimer`

### Functions

- `int main ()`

## 13.44.1 Macro Definition Documentation

### 13.44.1.1 CUBE

```
#define CUBE(
 x) ((x)*(x)*(x))
```

### 13.44.1.2 GFOPS

```
#define GFOPS(
 m,
 n,
 r,
 t) (2.0/3.0*CUBE(double(n)/1000.0) +2*m/1000.0*n/1000.0*double(r)/1000.0 - double(r)/1000.0*double(r)/1000.0*(m+n)/1000)/t
```

## 13.44.2 Typedef Documentation

### 13.44.2.1 TTimer

```
typedef Givaro::Timer TTimer
```

## 13.44.3 Function Documentation

### 13.44.3.1 main()

```
int main (
 void)
```

## 13.45 examples/pluq.C File Reference

```
#include <iostream>
#include <givaro/modular.h>
#include "fflas-ffpack/fflas-ffpack-config.h"
```



```
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/fflas_io.h"
```

## Functions

- int [main](#) (int argc, char \*\*argv)

### 13.45.1 Function Documentation

#### 13.45.1.1 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.46 rank.C File Reference

```
#include <iostream>
#include "fflas-ffpack/fflas-ffpack.h"
```

## Functions

- int [main](#) (int argc, char \*\*argv)

*This example computes the rank of a matrix over a defined finite field.*

### 13.46.1 Function Documentation

#### 13.46.1.1 main()

```
int main (
 int argc,
 char ** argv)
```

This example computes the rank of a matrix over a defined finite field.  
Outputs the rank.

## 13.47 solve.C File Reference

```
#include <iostream>
#include "fflas-ffpack/fflas-ffpack.h"
```

## Functions

- int [main](#) (int argc, char \*\*argv)

*This example solve the quare system defined by the input over a defined finite field.*

### 13.47.1 Function Documentation

#### 13.47.1.1 main()

```
int main (
 int argc,
 char ** argv)
```

This example solve the quare system defined by the input over a defined finite field.

## 13.48 checker\_charpoly.inl File Reference

```
#include "fflas-ffpack/ffpack/ffpack.h"
```

### Data Structures

- class [CheckerImplem\\_charpoly](#)< Field, Polynomial >
- class [CheckerImplem\\_charpoly](#)< Givaro::ZRing< Givaro::Integer >, Polynomial >

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

### Macros

- #define [\\_\\_FFLASFFPACK\\_checker\\_charpoly\\_INL](#)

### 13.48.1 Macro Definition Documentation

#### 13.48.1.1 \_\_FFLASFFPACK\_checker\_charpoly\_INL

```
#define __FFLASFFPACK_checker_charpoly_INL
```

## 13.49 checker\_det.inl File Reference

```
#include "fflas-ffpack/ffpack/ffpack.h"
```

### Data Structures

- class [CheckerImplem\\_Det](#)< Field >

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

### Macros

- #define [\\_\\_FFLASFFPACK\\_checker\\_det\\_INL](#)

### 13.49.1 Macro Definition Documentation

#### 13.49.1.1 \_\_FFLASFFPACK\_checker\_det\_INL

```
#define __FFLASFFPACK_checker_det_INL
```

## 13.50 checker\_empty.h File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
```

### Data Structures

- struct [Checker\\_Empty](#)< Field >

**Namespaces**

- namespace [FFLAS](#)

**13.51 checker\_fgemm.inl File Reference****Data Structures**

- class [CheckerImplem\\_fgemm< Field >](#)

**Namespaces**

- namespace [FFLAS](#)

**Macros**

- `#define __FFLASFFPACK_checker_fgemm_INL`

**13.51.1 Macro Definition Documentation****13.51.1.1 \_\_FFLASFFPACK\_checker\_fgemm\_INL**

```
#define __FFLASFFPACK_checker_fgemm_INL
```

**13.52 checker\_ftrsm.inl File Reference****Data Structures**

- class [CheckerImplem\\_ftrsm< Field >](#)

**Namespaces**

- namespace [FFLAS](#)

**Macros**

- `#define __FFLASFFPACK_checker_ftrsm_INL`

**13.52.1 Macro Definition Documentation****13.52.1.1 \_\_FFLASFFPACK\_checker\_ftrsm\_INL**

```
#define __FFLASFFPACK_checker_ftrsm_INL
```

**13.53 checker\_invert.inl File Reference****Data Structures**

- class [CheckerImplem\\_invert< Field >](#)

**Namespaces**

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

**Macros**

- `#define __FFLASFFPACK_checker_invert_INL`

### 13.53.1 Macro Definition Documentation

#### 13.53.1.1 \_\_FFLASFFPACK\_checker\_invert\_INL

```
#define __FFLASFFPACK_checker_invert_INL
```

## 13.54 checker\_pluq.inl File Reference

```
#include "fflas-ffpack/ffpack/ffpack.h"
#include "fflas-ffpack/utils/fflas_io.h"
```

### Data Structures

- class [CheckerImplem\\_PLUQ< Field >](#)

### Namespaces

- namespace [FFPACK](#)

*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

### Macros

- #define [\\_\\_FFLASFFPACK\\_checker\\_pluq\\_INL](#)

### 13.54.1 Macro Definition Documentation

#### 13.54.1.1 \_\_FFLASFFPACK\_checker\_pluq\_INL

```
#define __FFLASFFPACK_checker_pluq_INL
```

## 13.55 checkers.doxy File Reference

## 13.56 checkers\_fflas.h File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "checker_empty.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/fflas/fflas_enum.h"
#include "fflas-ffpack/utils/fflas_memory.h"
```

### Data Structures

- class [FailureFgemmCheck](#)
- class [FailureTrsmCheck](#)

### Namespaces

- namespace [FFLAS](#)

### Typedefs

- template<class [Field](#) >  
using [Checker\\_fgemm](#) = [FFLAS::Checker\\_Empty< Field >](#)
- template<class [Field](#) >  
using [Checker\\_ftsm](#) = [FFLAS::Checker\\_Empty< Field >](#)

## 13.57 checkers\_fflas.inl File Reference

```
#include "checker_fgemm.inl"
#include "checker_ftrsm.inl"
```

### Namespaces

- namespace [FFLAS](#)

### Macros

- `#define` [FFLASFFPACK\\_checkers\\_fflas\\_inl\\_H](#)

### Typedefs

- `template<class Field >`  
using [ForceCheck\\_fgemm](#) = [CheckerImplem\\_fgemm](#)< [Field](#) >
- `template<class Field >`  
using [ForceCheck\\_ftrsm](#) = [CheckerImplem\\_ftrsm](#)< [Field](#) >

### 13.57.1 Macro Definition Documentation

#### 13.57.1.1 FFLASFFPACK\_checkers\_fflas\_inl\_H

```
#define FFLASFFPACK_checkers_fflas_inl_H
```

## 13.58 checkers\_ffpack.h File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "checker_empty.h"
#include "fflas-ffpack/ffpack/ffpack.h"
```

### Data Structures

- class [FailurePLUQCheck](#)
- class [FailureDetCheck](#)
- class [FailureInvertCheck](#)
- class [FailureCharpolyCheck](#)

### Namespaces

- namespace [FFPACK](#)

*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

### Typedefs

- `template<class Field >`  
using [Checker\\_PLUQ](#) = [FFLAS::Checker\\_Empty](#)< [Field](#) >
- `template<class Field >`  
using [Checker\\_Det](#) = [FFLAS::Checker\\_Empty](#)< [Field](#) >
- `template<class Field >`  
using [Checker\\_invert](#) = [FFLAS::Checker\\_Empty](#)< [Field](#) >
- `template<class Field , class Polynomial >`  
using [Checker\\_charpoly](#) = [FFLAS::Checker\\_Empty](#)< [Field](#) >

## 13.59 checkers\_ffpack.inl File Reference

```
#include "checker_pluq.inl"
#include "checker_det.inl"
#include "checker_invert.inl"
#include "checker_charpoly.inl"
```

### Namespaces

- namespace [FFPACK](#)  
*Finite Field PACK Set of elimination based routines for dense linear algebra.*

### Macros

- #define [FFLASFFPACK\\_checkers\\_ffpack\\_inl\\_H](#)

### Typedefs

- template<class [Field](#) >  
using [ForceCheck\\_PLUQ](#) = [CheckerImplem\\_PLUQ](#)< [Field](#) >
- template<class [Field](#) >  
using [ForceCheck\\_Det](#) = [CheckerImplem\\_Det](#)< [Field](#) >
- template<class [Field](#) >  
using [ForceCheck\\_invert](#) = [CheckerImplem\\_invert](#)< [Field](#) >
- template<class [Field](#) , class Polynomial >  
using [ForceCheck\\_charpoly](#) = [CheckerImplem\\_charpoly](#)< [Field](#), Polynomial >

## 13.59.1 Macro Definition Documentation

### 13.59.1.1 FFLASFFPACK\_checkers\_ffpack\_inl\_H

```
#define FFLASFFPACK_checkers_ffpack_inl_H
```

## 13.60 config-blas.h File Reference

### Macros

- #define [CBLAS\\_INT](#) int
- #define [CBLAS\\_ENUM\\_DEFINED\\_H](#)
- #define [CBLAS\\_EXTERNALS](#)
- #define [blas\\_enum](#) enum

### Enumerations

- enum [CBLAS\\_ORDER](#) { [CblasRowMajor](#) =101 , [CblasColMajor](#) =102 }
- enum [CBLAS\\_TRANSPOSE](#) { [CblasNoTrans](#) =111 , [CblasTrans](#) =112 , [CblasConjTrans](#) =113 , [AtlasConj](#) =114 }
- enum [CBLAS\\_UPLO](#) { [CblasUpper](#) =121 , [CblasLower](#) =122 }
- enum [CBLAS\\_DIAG](#) { [CblasNonUnit](#) =131 , [CblasUnit](#) =132 }
- enum [CBLAS\\_SIDE](#) { [CblasLeft](#) =141 , [CblasRight](#) =142 }

### Functions

- void [daxpy\\_](#) (const int \*, const double \*, const double \*, const int \*, double \*, const int \*)
- void [saxpy\\_](#) (const int \*, const float \*, const float \*, const int \*, float \*, const int \*)
- double [ddot\\_](#) (const int \*, const double \*, const int \*, const double \*, const int \*)
- float [sdot\\_](#) (const int \*, const float \*, const int \*, const float \*, const int \*)

- double [dasum\\_](#) (const int \*, const double \*, const int \*)
- int [idamax\\_](#) (const int \*, const double \*, const int \*)
- double [dnrm2\\_](#) (const int \*, const double \*, const int \*)
- void [dgemv\\_](#) (const char \*, const int \*, const int \*, const double \*, const double \*, const int \*, const double \*, const int \*, const double \*, double \*, const int \*)
- void [sgemv\\_](#) (const char \*, const int \*, const int \*, const float \*, const float \*, const int \*, const float \*, const int \*, const float \*, float \*, const int \*)
- void [dger\\_](#) (const int \*, const int \*, const double \*, const double \*, const int \*, const double \*, const int \*, double \*, const int \*)
- void [sger\\_](#) (const int \*, const int \*, const float \*, const float \*, const int \*, const float \*, const int \*, float \*, const int \*)
- void [dcopy\\_](#) (const int \*, const double \*, const int \*, double \*, const int \*)
- void [scopy\\_](#) (const int \*, const float \*, const int \*, float \*, const int \*)
- void [dscal\\_](#) (const int \*, const double \*, double \*, const int \*)
- void [sscal\\_](#) (const int \*, const float \*, float \*, const int \*)
- void [dtrsm\\_](#) (const char \*, const char \*, const char \*, const char \*, const int \*, const int \*, const double \*, const double \*, const int \*, double \*, const int \*)
- void [strsm\\_](#) (const char \*, const char \*, const char \*, const char \*, const int \*, const int \*, const float \*, const float \*, const int \*, float \*, const int \*)
- void [dtrmm\\_](#) (const char \*, const char \*, const char \*, const char \*, const int \*, const int \*, const double \*, const double \*, const int \*, double \*, const int \*)
- void [strmm\\_](#) (const char \*, const char \*, const char \*, const char \*, const int \*, const int \*, const float \*, const float \*, const int \*, float \*, const int \*)
- void [sgemm\\_](#) (const char \*, const char \*, const int \*, const int \*, const int \*, const float \*, const float \*, const int \*, const float \*, const int \*, const float \*, float \*, const int \*)
- void [dgemm\\_](#) (const char \*, const char \*, const int \*, const int \*, const int \*, const double \*, const double \*, const int \*, const double \*, const int \*, const double \*, double \*, const int \*)
- void [cblas\\_dsyrk](#) (const enum [CBLAS\\_ORDER](#) Order, const enum [CBLAS\\_UPLO](#) Uplo, const enum [CBLAS\\_TRANSPOSE](#) Trans, const int N, const int K, const double alpha, const double \*A, const int lda, const double beta, double \*C, const int ldc)

## 13.60.1 Macro Definition Documentation

### 13.60.1.1 CBLAS\_INT

```
#define CBLAS_INT int
```

### 13.60.1.2 CBLAS\_ENUM\_DEFINED\_H

```
#define CBLAS_ENUM_DEFINED_H
```

### 13.60.1.3 CBLAS\_EXTERNALS

```
#define CBLAS_EXTERNALS
```

### 13.60.1.4 blas\_enum

```
#define blas_enum enum
```

## 13.60.2 Enumeration Type Documentation

### 13.60.2.1 CBLAS\_ORDER

```
enum CBLAS_ORDER
```

Enumerator

|               |  |
|---------------|--|
| CblasRowMajor |  |
| CblasColMajor |  |

### 13.60.2.2 CBLAS\_TRANSPOSE

enum [CBLAS\\_TRANSPOSE](#)

#### Enumerator

|                |  |
|----------------|--|
| CblasNoTrans   |  |
| CblasTrans     |  |
| CblasConjTrans |  |
| AtlasConj      |  |

### 13.60.2.3 CBLAS\_UPLO

enum [CBLAS\\_UPLO](#)

#### Enumerator

|            |  |
|------------|--|
| CblasUpper |  |
| CblasLower |  |

### 13.60.2.4 CBLAS\_DIAG

enum [CBLAS\\_DIAG](#)

#### Enumerator

|              |  |
|--------------|--|
| CblasNonUnit |  |
| CblasUnit    |  |

### 13.60.2.5 CBLAS\_SIDE

enum [CBLAS\\_SIDE](#)

#### Enumerator

|            |  |
|------------|--|
| CblasLeft  |  |
| CblasRight |  |

## 13.60.3 Function Documentation

### 13.60.3.1 daxpy\_()

```
void daxpy_ (
 const int * ,
 const double * ,
 const double * ,
 const int * ,
 double * ,
 const int *)
```

### 13.60.3.2 saxpy\_()

```
void saxpy_ (
 const int * ,
```



```
const float * ,
const float * ,
const int * ,
float * ,
const int *)
```

### 13.60.3.3 ddot\_()

```
double ddot_ (
 const int * ,
 const double * ,
 const int * ,
 const double * ,
 const int *)
```

### 13.60.3.4 sdot\_()

```
float sdot_ (
 const int * ,
 const float * ,
 const int * ,
 const float * ,
 const int *)
```

### 13.60.3.5 dasum\_()

```
double dasum_ (
 const int * ,
 const double * ,
 const int *)
```

### 13.60.3.6 idamax\_()

```
int idamax_ (
 const int * ,
 const double * ,
 const int *)
```

### 13.60.3.7 dnorm2\_()

```
double dnorm2_ (
 const int * ,
 const double * ,
 const int *)
```

### 13.60.3.8 dgemv\_()

```
void dgemv_ (
 const char * ,
 const int * ,
 const int * ,
 const double * ,
 const double * ,
 const int * ,
 const double * ,
 const int * ,
 const double * ,
 double * ,
 const int *)
```

**13.60.3.9 sgemv\_()**

```
void sgemv_ (
 const char * ,
 const int * ,
 const int * ,
 const float * ,
 const float * ,
 const int * ,
 const float * ,
 const int * ,
 const float * ,
 float * ,
 const int *)
```

**13.60.3.10 dger\_()**

```
void dger_ (
 const int * ,
 const int * ,
 const double * ,
 const double * ,
 const int * ,
 const double * ,
 const int * ,
 double * ,
 const int *)
```

**13.60.3.11 sger\_()**

```
void sger_ (
 const int * ,
 const int * ,
 const float * ,
 const float * ,
 const int * ,
 const float * ,
 const int * ,
 float * ,
 const int *)
```

**13.60.3.12 dcopy\_()**

```
void dcopy_ (
 const int * ,
 const double * ,
 const int * ,
 double * ,
 const int *)
```

**13.60.3.13 scopy\_()**

```
void scopy_ (
 const int * ,
 const float * ,
 const int * ,
 float * ,
 const int *)
```

**13.60.3.14 dscal\_()**

```
void dscal_ (
 const int * ,
 const double * ,
 double * ,
 const int *)
```

**13.60.3.15 sscal\_()**

```
void sscal_ (
 const int * ,
 const float * ,
 float * ,
 const int *)
```

**13.60.3.16 dtrsm\_()**

```
void dtrsm_ (
 const char * ,
 const char * ,
 const char * ,
 const char * ,
 const int * ,
 const int * ,
 const double * ,
 const double * ,
 const int * ,
 double * ,
 const int *)
```

**13.60.3.17 strsm\_()**

```
void strsm_ (
 const char * ,
 const char * ,
 const char * ,
 const char * ,
 const int * ,
 const int * ,
 const float * ,
 const float * ,
 const int * ,
 float * ,
 const int *)
```

**13.60.3.18 dtrmm\_()**

```
void dtrmm_ (
 const char * ,
 const char * ,
 const char * ,
 const char * ,
 const int * ,
 const int * ,
 const double * ,
 const double * ,
 const int * ,
 double * ,
 const int *)
```

### 13.60.3.19 strmm\_()

```
void strmm_ (
 const char * ,
 const char * ,
 const char * ,
 const char * ,
 const int * ,
 const int * ,
 const float * ,
 const float * ,
 const int * ,
 float * ,
 const int *)
```

### 13.60.3.20 sgemm\_()

```
void sgemm_ (
 const char * ,
 const char * ,
 const int * ,
 const int * ,
 const int * ,
 const float * ,
 const float * ,
 const int * ,
 const float * ,
 const int * ,
 const float * ,
 float * ,
 const int *)
```

### 13.60.3.21 dgemm\_()

```
void dgemm_ (
 const char * ,
 const char * ,
 const int * ,
 const int * ,
 const int * ,
 const double * ,
 const double * ,
 const int * ,
 const double * ,
 const int * ,
 const double * ,
 double * ,
 const int *)
```

### 13.60.3.22 cblas\_dsyrk()

```
void cblas_dsyrk (
 const enum CBLAS_ORDER Order,
 const enum CBLAS_UPLO Uplo,
 const enum CBLAS_TRANSPOSE Trans,
 const int N,
 const int K,
 const double alpha,
 const double * A,
```

```
const int lda,
const double beta,
double * C,
const int ldc)
```

## 13.61 fflas-ffpack-config.h File Reference

Defaults for optimised values.

```
#include "fflas-ffpack/config.h"
#include "fflas-ffpack/fflas-ffpack-thresholds.h"
#include "fflas-ffpack/fflas-ffpack-default-thresholds.h"
#include "givaro/givconfig.h"
```

### Macros

- #define [GCC\\_VERSION](#) (`__GNUC__ * 10000 + __GNUC_MINOR__ * 100 + __GNUC_PATCHLEVEL__`)

### 13.61.1 Detailed Description

Defaults for optimised values.

While `fflas-ffpack-optimize.h` is created by `configure` script, (either left blank or filled by optimiser), this file produces the defaults for the optimised values. If `fflas-ffpack-optimize.h` is not empty, then its values preceeds the defaults here.

### 13.61.2 Macro Definition Documentation

#### 13.61.2.1 GCC\_VERSION

```
#define GCC_VERSION (__GNUC__ * 10000 + __GNUC_MINOR__ * 100 + __GNUC_PATCHLEVEL__)
```

## 13.62 fflas-ffpack-default-thresholds.h File Reference

### Macros

- #define [\\_\\_FFLASFFPACK\\_WINOTHRESHOLD](#) 1000
- #define [\\_\\_FFLASFFPACK\\_WINOTHRESHOLD\\_FLT](#) 2000
- #define [\\_\\_FFLASFFPACK\\_WINOTHRESHOLD\\_BAL](#) 1000
- #define [\\_\\_FFLASFFPACK\\_WINOTHRESHOLD\\_BAL\\_FLT](#) 2000
- #define [\\_\\_FFLASFFPACK\\_PLUQ\\_THRESHOLD](#) 256
- #define [\\_\\_FFLASFFPACK\\_CHARPOLY\\_LUKrylov\\_ArithProg\\_THRESHOLD](#) 1000
- #define [\\_\\_FFLASFFPACK\\_CHARPOLY\\_Danilevskii\\_LUKrylov\\_THRESHOLD](#) 16
- #define [\\_\\_FFLASFFPACK\\_ARITHPROG\\_THRESHOLD](#) 30
- #define [\\_\\_FFLASFFPACK\\_FTRTRI\\_THRESHOLD](#) 32
- #define [\\_\\_FFLASFFPACK\\_FSYTRF\\_THRESHOLD](#) 64
- #define [\\_\\_FFLASFFPACK\\_FSYRK\\_THRESHOLD](#) 3000

### 13.62.1 Macro Definition Documentation

#### 13.62.1.1 \_\_FFLASFFPACK\_WINOTHRESHOLD

```
#define __FFLASFFPACK_WINOTHRESHOLD 1000
```

#### 13.62.1.2 \_\_FFLASFFPACK\_WINOTHRESHOLD\_FLT

```
#define __FFLASFFPACK_WINOTHRESHOLD_FLT 2000
```

**13.62.1.3 \_\_FFLASFFPACK\_WINOTHRESHOLD\_BAL**

```
#define __FFLASFFPACK_WINOTHRESHOLD_BAL 1000
```

**13.62.1.4 \_\_FFLASFFPACK\_WINOTHRESHOLD\_BAL\_FLT**

```
#define __FFLASFFPACK_WINOTHRESHOLD_BAL_FLT 2000
```

**13.62.1.5 \_\_FFLASFFPACK\_PLUQ\_THRESHOLD**

```
#define __FFLASFFPACK_PLUQ_THRESHOLD 256
```

**13.62.1.6 \_\_FFLASFFPACK\_CHARPOLY\_LUKrylov\_ArithProg\_THRESHOLD**

```
#define __FFLASFFPACK_CHARPOLY_LUKrylov_ArithProg_THRESHOLD 1000
```

**13.62.1.7 \_\_FFLASFFPACK\_CHARPOLY\_Danilevskii\_LUKrylov\_THRESHOLD**

```
#define __FFLASFFPACK_CHARPOLY_Danilevskii_LUKrylov_THRESHOLD 16
```

**13.62.1.8 \_\_FFLASFFPACK\_ARITHPROG\_THRESHOLD**

```
#define __FFLASFFPACK_ARITHPROG_THRESHOLD 30
```

**13.62.1.9 \_\_FFLASFFPACK\_FTRTRI\_THRESHOLD**

```
#define __FFLASFFPACK_FTRTRI_THRESHOLD 32
```

**13.62.1.10 \_\_FFLASFFPACK\_FSYTRF\_THRESHOLD**

```
#define __FFLASFFPACK_FSYTRF_THRESHOLD 64
```

**13.62.1.11 \_\_FFLASFFPACK\_FSYRK\_THRESHOLD**

```
#define __FFLASFFPACK_FSYRK_THRESHOLD 3000
```

**13.63 fflas-ffpack-thresholds.h File Reference****13.64 fflas-ffpack.doxy File Reference****13.65 fflas-ffpack.h File Reference**

Includes [FFLAS](#) and [FFPACK](#).

```
#include "fflas-ffpack/fflas-ffpack-config.h"
```

```
#include "fflas/fflas.h"
```

**13.65.1 Detailed Description**

Includes [FFLAS](#) and [FFPACK](#).

**13.66 fflas.doxy File Reference****13.67 fflas.h File Reference**

Finite Field Linear Algebra Subroutines

```

#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/config.h"
#include "fflas-ffpack/config-blas.h"
#include <cmath>
#include <cstring>
#include <float.h>
#include <algorithm>
#include "fflas_enum.h"
#include "fflas-ffpack/utils/fflas_memory.h"
#include "fflas-ffpack/paladin/parallel.h"
#include "fflas_level1.inl"
#include "fflas_level2.inl"
#include "fflas_level3.inl"
#include "fflas-ffpack/checkers/checkers_fflas.h"
#include "fflas_freduce.h"
#include "fflas_fadd.h"
#include "fflas_fscal.h"
#include "fflas_fassign.h"
#include "fflas_fgemm.inl"
#include "fflas_pfgemm.inl"
#include "fflas_fgemv.inl"
#include "fflas-ffpack/paladin/pfgemv.inl"
#include "fflas_freivalds.inl"
#include "fflas_fger.inl"
#include "fflas_fsyrk.inl"
#include "fflas_fsyrk_strassen.inl"
#include "fflas_fsyr2k.inl"
#include "fflas_ftrsm.inl"
#include "fflas_pftsm.inl"
#include "fflas_ftrmm.inl"
#include "fflas_ftrsv.inl"
#include "fflas_faxpy.inl"
#include "fflas_fdot.inl"
#include "fflas-ffpack/field/rns.h"
#include "fflas_fscal_mp.inl"
#include "fflas_freduce_mp.inl"
#include "fflas-ffpack/fflas/fflas_fger_mp.inl"
#include "fflas_fgemm/fgemm_classical_mp.inl"
#include "fflas_ftrsm_mp.inl"
#include "fflas_fgemv_mp.inl"
#include "fflas-ffpack/field/rns.inl"
#include "fflas-ffpack/paladin/fflas_plevel1.h"
#include "fflas_sparse.h"
#include "fflas-ffpack/checkers/checkers_fflas.inl"

```

## Macros

- `#define WINOTHRESHOLD __FFLASFFPACK_WINOTHRESHOLD`
- `#define DOUBLE_TO_FLOAT_CROSSOVER 800`

*Thresholds determining which floating point representation to use, depending on the cardinality of the finite field.*

## 13.67.1 Detailed Description

### Finite Field Linear Algebra Subroutines

## Author

Clément Pernet.

## 13.67.2 Macro Definition Documentation

### 13.67.2.1 WINOTHRESHOLD

```
#define WINOTHRESHOLD __FFLASFFPACK_WINOTHRESHOLD
```

### 13.67.2.2 DOUBLE\_TO\_FLOAT\_CROSSOVER

```
#define DOUBLE_TO_FLOAT_CROSSOVER 800
```

Thresholds determining which floating point representation to use, depending on the cardinality of the finite field. This is only used when the element representation is not a floating point type.

**Bug** to be benchmarked.

## 13.68 fflas\_bounds.inl File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/utils/flimits.h"
#include <givaro/udl.h>
#include <givaro/modular.h>
#include <givaro/modular-balanced.h>
```

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::Protected](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_bounds\\_INL](#)
- #define [FFLAS\\_INT\\_TYPE](#) uint64\_t

### Functions

- template<class [Field](#) >  
double [computeFactorClassic](#) (const [Field](#) &F)
- template<> double [computeFactorClassic](#) (const [Givaro::ModularBalanced](#)< double > &F)
- template<> double [computeFactorClassic](#) (const [Givaro::ModularBalanced](#)< float > &F)
- template<class [Field](#) >  
size\_t [DotProdBoundClassic](#) (const [Field](#) &F, const typename [Field::Element](#) &beta)
- [Givaro::Integer](#) [InfNorm](#) (const size\_t M, const size\_t N, const [Givaro::Integer](#) \*A, const size\_t lda)
- template<class [Field](#) >  
size\_t [TRSMBound](#) (const [Field](#) &)  
*TRSMBound.*
- template<class [Element](#) >  
size\_t [TRSMBound](#) (const [Givaro::Modular](#)< [Element](#) > &F)  
*Specialization for positive modular representation over float.*
- template<class [Element](#) >  
size\_t [TRSMBound](#) (const [Givaro::ModularBalanced](#)< [Element](#) > &F)  
*Specialization for balanced modular representation over double.*



## 13.68.1 Macro Definition Documentation

### 13.68.1.1 \_\_FFLASFFPACK\_fflas\_bounds\_INL

```
#define __FFLASFFPACK_fflas_bounds_INL
```

### 13.68.1.2 FFLAS\_INT\_TYPE

```
#define FFLAS_INT_TYPE uint64_t
```

## 13.69 fflas\_enum.h File Reference

```
#include <algorithm>
```

### Data Structures

- class [AreEqual< X, Y >](#)
- class [AreEqual< X, X >](#)

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::Protected](#)

### Enumerations

- enum [FFLAS\\_ORDER](#) { [FflasRowMajor](#) = 101 , [FflasColMajor](#) = 102 }  
*Storage by row or col ?*
- enum [FFLAS\\_TRANSPOSE](#) { [FflasNoTrans](#) = 111 , [FflasTrans](#) = 112 }  
*Is matrix transposed ?*
- enum [FFLAS\\_UPLO](#) { [FflasUpper](#) = 121 , [FflasLower](#) = 122 , [FflasLeftTri](#) = 123 , [FflasRightTri](#) = 124 }  
*Is triangular matrix's shape upper ?*
- enum [FFLAS\\_DIAG](#) { [FflasNonUnit](#) = 131 , [FflasUnit](#) = 132 }  
*Is the triangular matrix implicitly unit diagonal ?*
- enum [FFLAS\\_SIDE](#) { [FflasLeft](#) = 141 , [FflasRight](#) = 142 }  
*On what side ?*
- enum [FFLAS\\_BASE](#) { [FflasDouble](#) = 151 , [FflasFloat](#) = 152 , [FflasGeneric](#) = 153 }  
*FFLAS\_BASE determines the type of the element representation for Matrix Mult kernel.*

### Functions

- template<class T >  
const T & [min3](#) (const T &m, const T &n, const T &k)
- template<class T >  
const T & [max3](#) (const T &m, const T &n, const T &k)
- template<class T >  
const T & [min4](#) (const T &m, const T &n, const T &k, const T &l)
- template<class T >  
const T & [max4](#) (const T &m, const T &n, const T &k, const T &l)

## 13.70 fflas\_fadd.h File Reference

```
#include "fflas-ffpack/fflas/fflas_simd.h"
#include "fflas_fadd.inl"
```

## Data Structures

- struct [support\\_simd\\_add< T >](#)

## Namespaces

- namespace [FFLAS](#)

## Functions

- template<class [Field](#) >  
void [fadd](#) (const [Field](#) &F, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t inca, typename [Field::ConstElement\\_ptr](#) B, const size\_t incb, typename [Field::Element\\_ptr](#) C, const size\_t incc)
- template<class [Field](#) >  
void [faddin](#) (const [Field](#) &F, const size\_t N, typename [Field::ConstElement\\_ptr](#) B, const size\_t incb, typename [Field::Element\\_ptr](#) C, const size\_t incc)
- template<class [Field](#) >  
void [fsub](#) (const [Field](#) &F, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t inca, typename [Field::ConstElement\\_ptr](#) B, const size\_t incb, typename [Field::Element\\_ptr](#) C, const size\_t incc)
- template<class [Field](#) >  
void [fsubin](#) (const [Field](#) &F, const size\_t N, typename [Field::ConstElement\\_ptr](#) B, const size\_t incb, typename [Field::Element\\_ptr](#) C, const size\_t incc)
- template<class [Field](#) >  
void [fadd](#) (const [Field](#) &F, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t inca, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) B, const size\_t incb, typename [Field::Element\\_ptr](#) C, const size\_t incc)
- template<class [Field](#) >  
void [pfadd](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::Element\\_ptr](#) C, const size\_t ldc, const size\_t numths)
- template<class [Field](#) >  
void [pfsub](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::Element\\_ptr](#) C, const size\_t ldc, const size\_t numths)
- template<class [Field](#) >  
void [pfaddin](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::Element\\_ptr](#) C, const size\_t ldc, size\_t numths)
- template<class [Field](#) >  
void [pfsubin](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::Element\\_ptr](#) C, const size\_t ldc, size\_t numths)
- template<class [Field](#) >  
void [fadd](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::Element\\_ptr](#) C, const size\_t ldc)  
*fadd : matrix addition.*
- template<class [Field](#) >  
void [fsub](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::Element\\_ptr](#) C, const size\_t ldc)  
*fsub : matrix subtraction.*
- template<class [Field](#) >  
void [faddin](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::Element\\_ptr](#) C, const size\_t ldc)  
*faddin*
- template<class [Field](#) >  
void [faddin](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) uplo, const size\_t N, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::Element\\_ptr](#) C, const size\_t ldc)  
*fadding for symmetric matrices*

- template<class [Field](#) >  
void [fsubin](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::Element\\_ptr](#) C, const size\_t ldc)  
 $fsubin\ C = C - B$
- template<class [Field](#) >  
void [fadd](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::Element\\_ptr](#) C, const size\_t ldc)  
*fadd : matrix addition with scaling.*

## 13.71 fflas\_fadd.inl File Reference

```
#include "fflas-ffpack/fflas/fflas_simd.h"
```

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::vectorised](#)
- namespace [FFLAS::details](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fadd\\_INL](#)

### Functions

- template<class SimdT , class Element , bool positive>  
std::enable\_if< [is\\_simd](#)< SimdT >::value, void >::type [VEC\\_ADD](#) (SimdT &C, SimdT &A, SimdT &B, SimdT &Q, SimdT &T, SimdT &P, SimdT &NEGP, SimdT &MIN, SimdT &MAX)
- template<bool positive, class Element , class T1 , class T2 >  
std::enable\_if< [FFLAS::support\\_simd\\_add](#)< Element >::value, void >::type [addp](#) (Element \*T, const Element \*TA, const Element \*TB, size\_t n, Element p, T1 min\_, T2 max\_)
- template<class SimdT , class Element , bool positive>  
std::enable\_if< [is\\_simd](#)< SimdT >::value, void >::type [VEC\\_SUB](#) (SimdT &C, SimdT &A, SimdT &B, SimdT &Q, SimdT &T, SimdT &P, SimdT &NEGP, SimdT &MIN, SimdT &MAX)
- template<bool positive, class Element , class T1 , class T2 >  
std::enable\_if< [FFLAS::support\\_simd\\_add](#)< Element >::value, void >::type [subp](#) (Element \*T, const Element \*TA, const Element \*TB, const size\_t n, const Element p, const T1 min\_, const T2 max\_)
- template<class Element >  
std::enable\_if< [FFLAS::support\\_simd\\_add](#)< Element >::value, void >::type [add](#) (Element \*T, const Element \*TA, const Element \*TB, size\_t n)
- template<class Element >  
std::enable\_if< [FFLAS::support\\_simd\\_add](#)< Element >::value, void >::type [sub](#) (Element \*T, const Element \*TA, const Element \*TB, size\_t n)
- template<class [Field](#) , bool ADD>  
std::enable\_if< [FFLAS::support\\_simd\\_add](#)< typenameField::Element >::value, void >::type [fadd](#) (const [Field](#) &F, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t inca, typename [Field::ConstElement\\_ptr](#) B, const size\_t incb, typename [Field::Element\\_ptr](#) C, const size\_t incc, [FieldCategories::ModularTag](#))
- template<class [Field](#) , bool ADD>  
std::enable\_if<![FFLAS::support\\_simd\\_add](#)< typenameField::Element >::value, void >::type [fadd](#) (const [Field](#) &F, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t inca, typename [Field::ConstElement\\_ptr](#) B, const size\_t incb, typename [Field::Element\\_ptr](#) C, const size\_t incc, [FieldCategories::ModularTag](#))

- `template<class Field, bool ADD>`  
`void fadd (const Field &F, const size_t N, typename Field::ConstElement_ptr A, const size_t inca, type-`  
`name Field::ConstElement_ptr B, const size_t incb, typename Field::Element_ptr C, const size_t incc,`  
`FieldCategories::GenericTag)`
- `template<class Field, bool ADD>`  
`std::enable_if<!FFLAS::support_simd_add< typenameField::Element >::value, void >::type fadd`  
`(const Field &F, const size_t N, typename Field::ConstElement_ptr A, const size_t inca, typename`  
`Field::ConstElement_ptr B, const size_t incb, typename Field::Element_ptr C, const size_t incc,`  
`FieldCategories::UnparametricTag)`
- `template<class Field, bool ADD>`  
`std::enable_if< FFLAS::support_simd_add< typenameField::Element >::value, void >::type fadd`  
`(const Field &F, const size_t N, typename Field::ConstElement_ptr A, const size_t inca, typename`  
`Field::ConstElement_ptr B, const size_t incb, typename Field::Element_ptr C, const size_t incc,`  
`FieldCategories::UnparametricTag)`

### 13.71.1 Macro Definition Documentation

#### 13.71.1.1 \_\_FFLASFFPACK\_fadd\_INL

```
#define __FFLASFFPACK_fadd_INL
```

## 13.72 fflas\_fassign.h File Reference

```
#include "fflas_fassign.inl"
```

## 13.73 fflas\_fassign.inl File Reference

```
#include <string.h>
#include <givaro/modular.h>
#include <givaro/modular-balanced.h>
#include <givaro/zring.h>
#include "fflas-ffpack/utils/debug.h"
```

### Namespaces

- namespace [FFLAS](#)

### Macros

- `#define __FFLASFFPACK_fassign_INL`

### Functions

- `template<class Field >`  
`void fassign (const Field &F, const size_t N, typename Field::ConstElement_ptr Y, const size_t incY, typename`  
`Field::Element_ptr X, const size_t incX)`  

$$fassign : x \leftarrow y.$$
- `template<> void fassign (const Givaro::Modular< float > &F, const size_t N, const float *Y, const size_t incY,`  
`float *X, const size_t incX)`
- `template<> void fassign (const Givaro::ModularBalanced< float > &F, const size_t N, const float *Y, const`  
`size_t incY, float *X, const size_t incX)`
- `template<> void fassign (const Givaro::ZRing< float > &F, const size_t N, const float *Y, const size_t incY,`  
`float *X, const size_t incX)`
- `template<> void fassign (const Givaro::Modular< double > &F, const size_t N, const double *Y, const size_t incY,`  
`double *X, const size_t incX)`

- template<> void `fassign` (const `Givaro::ModularBalanced`< double > &F, const size\_t N, const double \*Y, const size\_t incY, double \*X, const size\_t incX)
- template<> void `fassign` (const `Givaro::ZRing`< double > &F, const size\_t N, const double \*Y, const size\_t incY, double \*X, const size\_t incX)
- template<class `Field` >  
void `fassign` (const `Field` &F, const size\_t m, const size\_t n, typename `Field::ConstElement_ptr` B, const size\_t ldb, typename `Field::Element_ptr` A, const size\_t lda)

*fassign : A ← B.*

## 13.73.1 Macro Definition Documentation

### 13.73.1.1 \_\_FFLASFFPACK\_fassign\_INL

```
#define __FFLASFFPACK_fassign_INL
```

## 13.74 fflas\_faxpy.inl File Reference

### Namespaces

- namespace `FFLAS`
- namespace `FFLAS::vectorised`
- namespace `FFLAS::vectorised::unswitch`
- namespace `FFLAS::details`

### Macros

- #define `__FFLASFFPACK_faxpy_INL`

### Functions

- template<class `Field` >  
std::enable\_if<!`FFLAS::support_simd_mod`< typename `Field::Element` >::value &&`FFLAS::support_fast_mod`< typename `Field::Element` >::value, void >::type `axypy` (const `Field` &F, const typename `Field::Element` a, typename `Field::ConstElement_ptr` X, typename `Field::Element_ptr` Y, const size\_t n, `HelperMod`< `Field` > &H)
- template<class `Field` >  
std::enable\_if< `FFLAS::support_fast_mod`< typename `Field::Element` >::value, void >::type `axypy` (const `Field` &F, const typename `Field::Element` a, typename `Field::ConstElement_ptr` X, typename `Field::Element_ptr` Y, const size\_t n, const size\_t incX, const size\_t incY, `HelperMod`< `Field` > &H)
- template<class `Field` >  
std::enable\_if< `FFLAS::support_fast_mod`< typename `Field::Element` >::value, void >::type `axypy` (const `Field` &F, const typename `Field::Element` a, typename `Field::ConstElement_ptr` X, typename `Field::Element_ptr` Y, const size\_t n)
- template<class `Field` >  
std::enable\_if< `FFLAS::support_fast_mod`< typename `Field::Element` >::value, void >::type `axypy` (const `Field` &F, const typename `Field::Element` a, typename `Field::ConstElement_ptr` X, typename `Field::Element_ptr` Y, const size\_t n, const size\_t incX, const size\_t incY)
- template<class `Field` >  
std::enable\_if< `FFLAS::support_fast_mod`< typename `Field::Element` >::value, void >::type `faxpy` (const `Field` &F, const size\_t N, const typename `Field::Element` a, typename `Field::ConstElement_ptr` X, const size\_t incX, typename `Field::Element_ptr` Y, const size\_t incY, `FieldCategories::ModularTag`)
- template<class `Field`, class `FC` >  
void `faxpy` (const `Field` &F, const size\_t N, const typename `Field::Element` a, typename `Field::ConstElement_ptr` X, const size\_t incX, typename `Field::Element_ptr` Y, const size\_t incY, `FC`)
- template<class `Field` >  
void `faxpy` (const `Field` &F, const size\_t N, const typename `Field::Element` alpha, typename `Field::ConstElement_ptr` X, const size\_t incX, typename `Field::Element_ptr` Y, const size\_t incY)

- $\text{faxpy} : y \leftarrow \alpha \cdot x + y.$
- `template<> void faxpy (const Givaro::DoubleDomain &, const size_t N, const Givaro::DoubleDomain::Element a, Givaro::DoubleDomain::ConstElement_ptr x, const size_t incx, Givaro::DoubleDomain::Element_ptr y, const size_t incy)`
  - `template<> void faxpy (const Givaro::FloatDomain &, const size_t N, const Givaro::FloatDomain::Element a, Givaro::FloatDomain::ConstElement_ptr x, const size_t incx, Givaro::FloatDomain::Element_ptr y, const size_t incy)`
  - `template<class Field >`  
`void faxpy (const Field &F, const size_t m, const size_t n, const typename Field::Element alpha, typename Field::ConstElement_ptr X, const size_t ldx, typename Field::Element_ptr Y, const size_t ldy)`  
 $\text{faxpy} : y \leftarrow \alpha \cdot x + y.$

### 13.74.1 Macro Definition Documentation

#### 13.74.1.1 \_\_FFLASFFPACK\_faxpy\_INL

```
#define __FFLASFFPACK_faxpy_INL
```

## 13.75 fflas\_fdot.inl File Reference

```
#include "fflas-ffpack/fflas/fflas_helpers.inl"
```

### Namespaces

- namespace `FFLAS`

### Macros

- `#define __FFLASFFPACK_fdot_INL`

### Functions

- `template<class Field >`  
`Field::Element fdot (const Field &F, const size_t N, typename Field::ConstElement_ptr x, const size_t incx, typename Field::ConstElement_ptr y, const size_t incy, ModeCategories::DefaultTag &MT)`
- `template<class Field >`  
`Field::Element fdot (const Field &F, const size_t N, typename Field::ConstElement_ptr x, const size_t incx, typename Field::ConstElement_ptr y, const size_t incy, ModeCategories::DelayedTag &MT)`
- `template<> Givaro::DoubleDomain::Element fdot (const Givaro::DoubleDomain &, const size_t N, Givaro::DoubleDomain::ConstElement_ptr x, const size_t incx, Givaro::DoubleDomain::ConstElement_ptr y, const size_t incy, ModeCategories::DefaultTag &MT)`
- `template<> Givaro::FloatDomain::Element fdot (const Givaro::FloatDomain &, const size_t N, Givaro::FloatDomain::ConstElement_ptr x, const size_t incx, Givaro::FloatDomain::ConstElement_ptr y, const size_t incy, ModeCategories::DefaultTag &MT)`
- `template<class Field, class T >`  
`Field::Element fdot (const Field &F, const size_t N, typename Field::ConstElement_ptr x, const size_t incx, typename Field::ConstElement_ptr y, const size_t incy, ModeCategories::ConvertTo< T > &MT)`
- `template<class Field >`  
`Field::Element fdot (const Field &F, const size_t N, typename Field::ConstElement_ptr x, const size_t incx, typename Field::ConstElement_ptr y, const size_t incy, ModeCategories::DefaultBoundedTag &dbt)`
- `template<class Field >`  
`Field::Element fdot (const Field &F, const size_t N, typename Field::ConstElement_ptr x, const size_t incx, typename Field::ConstElement_ptr y, const size_t incy, const ParSeqHelper::Sequential seq)`
- `template<class Field >`  
`Field::Element fdot (const Field &F, const size_t N, typename Field::ConstElement_ptr X, const size_t incX, typename Field::ConstElement_ptr Y, const size_t incY)`  
 $\text{fdot} : \text{dot product } x^T y.$

## 13.75.1 Macro Definition Documentation

### 13.75.1.1 \_\_FFLASFFPACK\_fdot\_INL

```
#define __FFLASFFPACK_fdot_INL
```

## 13.76 fflas\_fgemm.inl File Reference

```
#include <givaro/modular.h>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/utils/debug.h"
#include "fflas_fgemm/fgemm_classical.inl"
#include "fflas_fgemm/fgemm_winograd.inl"
```

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::Protected](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fgemm\\_INL](#)

### Functions

- template<class NewField , class [Field](#) , class FieldMode >  
[Field::Element\\_ptr fgemm\\_convert](#) (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const [FFLAS\\_TRANSPOSE](#) tb, const size\_t m, const size\_t n, const size\_t k, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc, [MMHelper](#)< [Field](#), [MMHelperAlgo::Winograd](#), FieldMode > &H)
- template<class [Field](#) , class Element , class AlgoT , class ParSeqTrait >  
bool [NeedPreAddReduction](#) (Element &Outmin, Element &Outmax, Element &Op1min, Element &Op1max, Element &Op2min, Element &Op2max, [MMHelper](#)< [Field](#), AlgoT, [ModeCategories::LazyTag](#), ParSeqTrait > &WH)
- template<class [Field](#) , class Element , class AlgoT , class ModeT , class ParSeqTrait >  
bool [NeedPreAddReduction](#) (Element &Outmin, Element &Outmax, Element &Op1min, Element &Op1max, Element &Op2min, Element &Op2max, [MMHelper](#)< [Field](#), AlgoT, ModeT, ParSeqTrait > &WH)
- template<class [Field](#) , class Element , class AlgoT , class ParSeqTrait >  
bool [NeedPreSubReduction](#) (Element &Outmin, Element &Outmax, Element &Op1min, Element &Op1max, Element &Op2min, Element &Op2max, [MMHelper](#)< [Field](#), AlgoT, [ModeCategories::LazyTag](#), ParSeqTrait > &WH)
- template<class [Field](#) , class Element , class AlgoT , class ModeT , class ParSeqTrait >  
bool [NeedPreSubReduction](#) (Element &Outmin, Element &Outmax, Element &Op1min, Element &Op1max, Element &Op2min, Element &Op2max, [MMHelper](#)< [Field](#), AlgoT, ModeT, ParSeqTrait > &WH)
- template<class [Field](#) , class Element , class AlgoT , class ParSeqTrait >  
bool [NeedDoublePreAddReduction](#) (Element &Outmin, Element &Outmax, Element &Op1min, Element &Op1max, Element &Op2min, Element &Op2max, Element beta, [MMHelper](#)< [Field](#), AlgoT, [ModeCategories::LazyTag](#), ParSeqTrait > &WH)
- template<class [Field](#) , class Element , class AlgoT , class ModeT , class ParSeqTrait >  
bool [NeedDoublePreAddReduction](#) (Element &Outmin, Element &Outmax, Element &Op1min, Element &Op1max, Element &Op2min, Element &Op2max, Element beta, [MMHelper](#)< [Field](#), AlgoT, ModeT, ParSeqTrait > &WH)
- template<class [Field](#) , class AlgoT , class ParSeqTrait >  
void [ScalAndReduce](#) (const [Field](#) &F, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::Element\\_ptr](#) X, const size\_t incX, const [MMHelper](#)< [Field](#), AlgoT, [ModeCategories::LazyTag](#), ParSeqTrait > &H)

- `template<class Field , class AlgoT , class ParSeqTrait >`  
`void ScalAndReduce (const Field &F, const size_t M, const size_t N, const typename Field::Element alpha,`  
`typename Field::Element_ptr A, const size_t lda, const MMHelper< Field, AlgoT, ModeCategories::LazyTag,`  
`ParSeqTrait > &H)`
- `template<class Field >`  
`Field::Element_ptr fgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE`  
`tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, typename`  
`Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb,`  
`const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper<`  
`Field, MMHelperAlgo::Winograd, ModeCategories::ConvertTo< ElementCategories::MachineFloatTag >,`  
`ParSeqHelper::Sequential > &H)`
- `template<typename Field >`  
`Field::Element_ptr fgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE`  
`tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, type-`  
`name Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t`  
`ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, const`  
`ParSeqHelper::Sequential seq)`
- `template<typename Field , class Cut , class Param >`  
`Field::Element_ptr fgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE`  
`tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, type-`  
`name Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t`  
`ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, const`  
`ParSeqHelper::Parallel< Cut, Param > par)`
- `template<typename Field >`  
`Field::Element_ptr fgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE`  
`tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, typename`  
`Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, const`  
`typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc)`  

*fgemm: Field **GE**neral **M**atrix **M**ultiply.*
- `template<typename Field , class ModeT , class ParSeq >`  
`Field::Element_ptr fgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE`  
`tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, typename`  
`Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb,`  
`const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field,`  
`MMHelperAlgo::Auto, ModeT, ParSeq > &H)`
- `template<class Field >`  
`Field::Element_ptr fgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE`  
`tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, typename`  
`Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb,`  
`const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field,`  
`MMHelperAlgo::Winograd, ModeCategories::DelayedTag, ParSeqHelper::Sequential > &H)`
- `template<class Field >`  
`Field::Element_ptr fsquare (const Field &F, const FFLAS_TRANSPOSE ta, const size_t n, const typename`  
`Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, const typename Field::Element`  
`beta, typename Field::Element_ptr C, const size_t ldc)`  

*fsquare: Squares a matrix.*
- `template<class Field >`  
`Field::Element_ptr fsquareCommon (const Field &F, const FFLAS_TRANSPOSE ta, const size_t n, const`  
`typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, const typename`  
`Field::Element beta, typename Field::Element_ptr C, const size_t ldc)`
- `template<> double * fsquare (const Givaro::ModularBalanced< double > &F, const FFLAS_TRANSPOSE`  
`ta, const size_t n, const double alpha, const double *A, const size_t lda, const double beta, double *C, const`  
`size_t ldc)`
- `template<> float * fsquare (const Givaro::ModularBalanced< float > &F, const FFLAS_TRANSPOSE ta,`  
`const size_t n, const float alpha, const float *A, const size_t lda, const float beta, float *C, const size_t ldc)`
- `template<> double * fsquare (const Givaro::Modular< double > &F, const FFLAS_TRANSPOSE ta, const`  
`size_t n, const double alpha, const double *A, const size_t lda, const double beta, double *C, const size_t`  
`ldc)`



- `template<> float * fsquare (const Givaro::Modular< float > &F, const FFLAS_TRANSPOSE ta, const size_t n, const float alpha, const float *A, const size_t lda, const float beta, float *C, const size_t ldc)`

## 13.76.1 Macro Definition Documentation

### 13.76.1.1 \_\_FFLASFFPACK\_fgemm\_INL

```
#define __FFLASFFPACK_fgemm_INL
```

## 13.77 fgemm\_classical.inl File Reference

## 13.78 fgemm\_classical\_mp.inl File Reference

matrix multiplication with multiprecision input (either over Z or over Z/pZ)

```
#include <givaro/modular-integer.h>
#include <givaro/zring.h>
#include "fflas-ffpack/field/rns-double.h"
#include "fflas-ffpack/field/rns-integer.h"
#include "fflas-ffpack/field/rns-integer-mod.h"
#include "fflas-ffpack/field/field-traits.h"
#include "fflas-ffpack/fflas/fflas_helpers.inl"
#include "fflas-ffpack/fflas/fflas_bounds.inl"
```

### Data Structures

- struct `MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeqTrait >`
- struct `MMHelper< FFPACK::RNSInteger< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >`
- struct `MMHelper< FFPACK::RNSIntegerMod< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >`

### Namespaces

- namespace `FFLAS`

### Macros

- `#define __FFPACK_fgemm_classical_INL`

### Functions

- `template<typename RNS, typename ParSeqTrait > FFPACK::RNSInteger< RNS >::Element_ptr fgemm (const FFPACK::RNSInteger< RNS > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename FFPACK::RNSInteger< RNS >::Element alpha, typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Ad, const size_t lda, typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Bd, const size_t ldb, const typename FFPACK::RNSInteger< RNS >::Element beta, typename FFPACK::RNSInteger< RNS >::Element_ptr Cd, const size_t ldc, MMHelper< FFPACK::RNSInteger< RNS >, MMHelperAlgo::Classic, ModeCategories::DefaultTag, ParSeqHelper::Compose< ParSeqHelper::Sequential, ParSeqTrait > > &H)`
- `template<typename RNS > FFPACK::RNSInteger< RNS >::Element_ptr fgemm (const FFPACK::RNSInteger< RNS > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename FFPACK::RNSInteger< RNS >::Element alpha, typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Ad, const size_t lda, typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Bd, const size_t ldb, const typename FFPACK::RNSInteger< RNS >::Element beta, typename FFPACK::RNSInteger< RNS >::Element_ptr Cd, const size_t ldc, MMHelper< FFPACK::RNSInteger< RNS >, MMHelperAlgo::Classic, ModeCategories::DefaultTag, ParSeqHelper::Sequential > &H)`

- `template<typename RNS , typename ParSeqTrait >`  
`FFPACK::RNSInteger< RNS >::Element_ptr fgemm (const FFPACK::RNSInteger< RNS > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename FFPACK::RNSInteger< RNS >::Element alpha, typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Ad, const size_t lda, typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Bd, const size_t ldb, const typename FFPACK::RNSInteger< RNS >::Element beta, typename FFPACK::RNSInteger< RNS >::Element_ptr Cd, const size_t ldc, MMHelper< FFPACK::RNSInteger< RNS >, MMHelperAlgo::Classic, ModeCategories::DefaultTag, ParSeqHelper::Compose< ParSeqHelper::Parallel< CuttingStrategy::RNSModulus, StrategyParameter::Threads >, ParSeqTrait > > &H)`
- `template<typename RNS , typename Cut , typename Param >`  
`FFPACK::RNSInteger< RNS >::Element_ptr fgemm (const FFPACK::RNSInteger< RNS > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename FFPACK::RNSInteger< RNS >::Element alpha, typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Ad, const size_t lda, typename FFPACK::RNSInteger< RNS >::ConstElement_ptr Bd, const size_t ldb, const typename FFPACK::RNSInteger< RNS >::Element beta, typename FFPACK::RNSInteger< RNS >::Element_ptr Cd, const size_t ldc, MMHelper< FFPACK::RNSInteger< RNS >, MMHelperAlgo::Classic, ModeCategories::DefaultTag, ParSeqHelper::Parallel< Cut, Param > > &H)`
- `template<class ParSeq >`  
`Givaro::Integer * fgemm (const Givaro::ZRing< Givaro::Integer > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const Givaro::Integer alpha, const Givaro::Integer *A, const size_t lda, const Givaro::Integer *B, const size_t ldb, Givaro::Integer beta, Givaro::Integer *C, const size_t ldc, MMHelper< Givaro::ZRing< Givaro::Integer >, MMHelperAlgo::Classic, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeq > &H)`
- `template<typename RNS , class ModeT >`  
`RNS::Element_ptr fgemm (const FFPACK::RNSInteger< RNS > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename RNS::Element alpha, typename RNS::ConstElement_ptr Ad, const size_t lda, typename RNS::ConstElement_ptr Bd, const size_t ldb, const typename RNS::Element beta, typename RNS::Element_ptr Cd, const size_t ldc, MMHelper< FFPACK::RNSInteger< RNS >, MMHelperAlgo::Winograd, ModeT, ParSeqHelper::Sequential > &H)`
- `template<typename RNS >`  
`RNS::Element_ptr fgemm (const FFPACK::RNSIntegerMod< RNS > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const typename RNS::Element alpha, typename RNS::ConstElement_ptr Ad, const size_t lda, typename RNS::ConstElement_ptr Bd, const size_t ldb, const typename RNS::Element beta, typename RNS::Element_ptr Cd, const size_t ldc, MMHelper< FFPACK::RNSIntegerMod< RNS >, MMHelperAlgo::Winograd > &H)`
- `Givaro::Integer * fgemm (const Givaro::Modular< Givaro::Integer > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const Givaro::Integer alpha, const Givaro::Integer *A, const size_t lda, const Givaro::Integer *B, const size_t ldb, const Givaro::Integer beta, Givaro::Integer *C, const size_t ldc, MMHelper< Givaro::Modular< Givaro::Integer >, MMHelperAlgo::Classic, ModeCategories::ConvertTo< ElementCategories::RNSElementTag > > &H)`
- `template<class ParSeq >`  
`Givaro::Integer * fgemm (const Givaro::Modular< Givaro::Integer > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const Givaro::Integer alpha, const Givaro::Integer *A, const size_t lda, const Givaro::Integer *B, const size_t ldb, const Givaro::Integer beta, Givaro::Integer *C, const size_t ldc, MMHelper< Givaro::Modular< Givaro::Integer >, MMHelperAlgo::Auto, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeq > &H)`
- `template<size_t K1, size_t K2, class ParSeq >`  
`Reclnt::ruint< K1 > * fgemm (const Givaro::Modular< Reclnt::ruint< K1 >, Reclnt::ruint< K2 > > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const Reclnt::ruint< K1 > alpha, const Reclnt::ruint< K1 > *A, const size_t lda, const Reclnt::ruint< K1 > *B, const size_t ldb, Reclnt::ruint< K1 > beta, Reclnt::ruint< K1 > *C, const size_t ldc, MMHelper< Givaro::Modular< Reclnt::ruint< K1 >, Reclnt::ruint< K2 > >, MMHelperAlgo::Classic, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeq > &H)`

### 13.78.1 Detailed Description

matrix multiplication with multiprecision input (either over  $\mathbb{Z}$  or over  $\mathbb{Z}/p\mathbb{Z}$ )

## 13.78.2 Macro Definition Documentation

### 13.78.2.1 \_\_FFPACK\_fgemm\_classical\_INL

```
#define __FFPACK_fgemm_classical_INL
```

## 13.79 fgemm\_winograd.inl File Reference

```
#include <stdint.h>
#include <givaro/modular.h>
#include <givaro/zring.h>
#include "fgemm_classical.inl"
#include "schedule_winograd.inl"
#include "schedule_winograd_acc.inl"
#include "schedule_winograd_acc_ip.inl"
#include "schedule_winograd_ip.inl"
#include "fflas-ffpack/fflas-ffpack-config.h"
```

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::Protected](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_fflas\\_fgemm\\_winograd\\_INL](#)
- #define [NEWWINO](#)

### Functions

- template<class [Field](#) >  
int [WinogradThreshold](#) (const [Field](#) &F)  
*Computes the number of recursive levels to perform.*
- template<> int [WinogradThreshold](#) (const Givaro::Modular< float > &F)
- template<> int [WinogradThreshold](#) (const [Givaro::ModularBalanced](#)< double > &F)
- template<> int [WinogradThreshold](#) (const [Givaro::ModularBalanced](#)< float > &F)
- template<class [Field](#) >  
int [WinogradSteps](#) (const [Field](#) &F, const size\_t &m)  
*Computes the number of recursive levels to perform.*
- template<class [Field](#) , class [FieldMode](#) >  
void [DynamicPeeling](#) (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const [FFLAS\\_TRANSPOSE](#) tb, const size\_t m, const size\_t n, const size\_t k, const size\_t mr, const size\_t nr, const size\_t kr, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc, [MMHelper](#)< [Field](#), [MMHelperAlgo::Winograd](#), [FieldMode](#) > &H, const typename [MMHelper](#)< [Field](#), [MMHelperAlgo::Winograd](#), [FieldMode](#) >::DelayedField::Element Cmin, const typename [MMHelper](#)< [Field](#), [MMHelperAlgo::Winograd](#), [FieldMode](#) >::DelayedField::Element Cmax)
- template<class [Field](#) , class [FieldMode](#) >  
void [DynamicPeeling2](#) (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const [FFLAS\\_TRANSPOSE](#) tb, const size\_t m, const size\_t n, const size\_t k, const size\_t mr, const size\_t nr, const size\_t kr, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc, [MMHelper](#)< [Field](#), [MMHelperAlgo::Winograd](#), [FieldMode](#) > &H, const typename [MMHelper](#)< [Field](#), [MMHelperAlgo::Winograd](#), [FieldMode](#) >::DelayedField::Element Cmin, const typename [MMHelper](#)< [Field](#), [MMHelperAlgo::Winograd](#), [FieldMode](#) >::DelayedField::Element Cmax)

- `template<class Field , class FieldMode >`  
`void WinogradCalc (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb,`  
`const size_t mr, const size_t nr, const size_t kr, const typename Field::Element alpha, typename`  
`Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb,`  
`const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field,`  
`MMHelperAlgo::Winograd, FieldMode > &H)`
- `template<class Field , class ModeT >`  
`Field::Element_ptr fgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE`  
`tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, typename`  
`Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb,`  
`const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field,`  
`MMHelperAlgo::Winograd, ModeT > &H)`
- `template<class Field , class ModeT , class Cut , class Param >`  
`Field::Element_ptr fgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE`  
`tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, typename`  
`Field::ConstElement_ptr A, const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb,`  
`const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field,`  
`MMHelperAlgo::WinogradPar, ModeT, ParSeqHelper::Parallel< Cut, Param > > &H)`

## 13.79.1 Macro Definition Documentation

### 13.79.1.1 \_\_FFLASFFPACK\_fflas\_fflas\_fgemm\_winograd\_INL

```
#define __FFLASFFPACK_fflas_fflas_fgemm_winograd_INL
```

### 13.79.1.2 NEWWINO

```
#define NEWWINO
```

## 13.80 matmul.doxy File Reference

## 13.81 schedule\_bini.inl File Reference

Bini implementation.

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::BLAS3](#)

### Macros

- `#define __FFLASFFPACK_fgemm_bini_INL`

### Functions

- `template<class Field >`  
`void Bini (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t mr,`  
`const size_t nr, const size_t kr, const typename Field::Element alpha, const typename Field::Element_ptr A,`  
`const size_t lda, const typename Field::Element_ptr B, const size_t ldb, const typename Field::Element beta,`  
`typename Field::Element_ptr C, const size_t ldc, const size_t kmax, const size_t w, const FFLAS_BASE`  
`base, const size_t rec_level)`

### 13.81.1 Detailed Description

Bini implementation.

## 13.81.2 Macro Definition Documentation

### 13.81.2.1 \_\_FFLASFFPACK\_fgemm\_bini\_INL

```
#define __FFLASFFPACK_fgemm_bini_INL
```

## 13.82 schedule\_winograd.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::BLAS3](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fgemm\\_winograd\\_INL](#)

### Functions

- template<class [Field](#) , class [FieldTrait](#) , class [Strat](#) , class [Param](#) >  
[Field::Element\\_ptr](#) [WinoPar](#) (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const [FFLAS\\_TRANSPOSE](#) tb, const size\_t mr, const size\_t nr, const size\_t kr, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc, [MMHelper](#)< [Field](#), [MMHelperAlgo::WinogradPar](#), [FieldTrait](#), [ParSeqHelper::Parallel](#)< [Strat](#), [Param](#) > > &WH)
- template<class [Field](#) , class [FieldTrait](#) >  
void [Winograd](#) (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const [FFLAS\\_TRANSPOSE](#) tb, const size\_t mr, const size\_t nr, const size\_t kr, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc, [MMHelper](#)< [Field](#), [MMHelperAlgo::Winograd](#), [FieldTrait](#)> &WH)

## 13.82.1 Macro Definition Documentation

### 13.82.1.1 \_\_FFLASFFPACK\_fgemm\_winograd\_INL

```
#define __FFLASFFPACK_fgemm_winograd_INL
```

## 13.83 schedule\_winograd\_acc.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::BLAS3](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fgemm\\_winograd\\_acc\\_INL](#)

### Functions

- template<class [Field](#) , class [FieldTrait](#) >  
void [WinogradAcc\\_3\\_23](#) (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const [FFLAS\\_TRANSPOSE](#) tb, const size\_t mr, const size\_t nr, const size\_t kr, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc, [MMHelper](#)< [Field](#), [MMHelperAlgo::Winograd](#), [FieldTrait](#) > &WH)

- `template<class Field , class FieldTrait >`  
`void WinogradAcc_3_21 (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t mr, const size_t nr, const size_t kr, const typename Field::Element alpha, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > &WH)`
- `template<class Field , class FieldTrait >`  
`void WinogradAcc_2_24 (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t mr, const size_t nr, const size_t kr, const typename Field::Element alpha, const typename Field::Element_ptr A, const size_t lda, const typename Field::Element_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > &WH)`
- `template<class Field , class FieldTrait >`  
`void WinogradAcc_2_27 (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t mr, const size_t nr, const size_t kr, const typename Field::Element alpha, const typename Field::Element_ptr A, const size_t lda, const typename Field::Element_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > &WH)`

### 13.83.1 Macro Definition Documentation

#### 13.83.1.1 \_\_FFLASFFPACK\_fgemm\_winograd\_acc\_INL

```
#define __FFLASFFPACK_fgemm_winograd_acc_INL
```

## 13.84 schedule\_winograd\_acc\_ip.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::BLAS3](#)

### Macros

- `#define __FFLASFFPACK_fgemm_winograd_acc_ip_INL`

### Functions

- `template<class Field , class FieldTrait >`  
`void WinogradAcc_LR (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t mr, const size_t nr, const size_t kr, const typename Field::Element alpha, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, const MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > &WH)`
- `template<class Field , class FieldTrait >`  
`void WinogradAcc_R_S (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t mr, const size_t nr, const size_t kr, const typename Field::Element alpha, const typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, const MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > &WH)`
- `template<class Field , class FieldTrait >`  
`void WinogradAcc_L_S (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t mr, const size_t nr, const size_t kr, const typename Field::Element alpha, typename Field::Element_ptr A, const size_t lda, const typename Field::Element_ptr B, const size_t ldb, const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, const MMHelper< Field, MMHelperAlgo::Winograd, FieldTrait > &WH)`

## 13.84.1 Macro Definition Documentation

### 13.84.1.1 \_\_FFLASFFPACK\_fgemm\_winograd\_acc\_ip\_INL

```
#define __FFLASFFPACK_fgemm_winograd_acc_ip_INL
```

## 13.85 schedule\_winograd\_ip.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::BLAS3](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fgemm\\_winograd\\_ip\\_INL](#)

### Functions

- template<class [Field](#) , class FieldTrait >  
void [Winograd\\_LR\\_S](#) (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const [FFLAS\\_TRANSPOSE](#) tb, const size\_t mr, const size\_t nr, const size\_t kr, const typename [Field::Element](#) alpha, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) B, const size\_t ldb, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc, const [MMHelper](#)< [Field](#), [MMHelperAlgo::Winograd](#), Field↵Trait > &WH)
- template<class [Field](#) , class FieldTrait >  
void [Winograd\\_L\\_S](#) (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const [FFLAS\\_TRANSPOSE](#) tb, const size\_t mr, const size\_t nr, const size\_t kr, const typename [Field::Element](#) alpha, typename [Field::Element\\_ptr](#) A, const size\_t lda, const typename [Field::Element\\_ptr](#) B, const size\_t ldb, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc, const [MMHelper](#)< [Field](#), [MMHelperAlgo::Winograd](#), FieldTrait > &WH)
- template<class [Field](#) , class FieldTrait >  
void [Winograd\\_R\\_S](#) (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) ta, const [FFLAS\\_TRANSPOSE](#) tb, const size\_t mr, const size\_t nr, const size\_t kr, const typename [Field::Element](#) alpha, const typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) B, const size\_t ldb, const type-  
name [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc, const [MMHelper](#)< [Field](#),  
[MMHelperAlgo::Winograd](#), FieldTrait > &WH)

## 13.85.1 Macro Definition Documentation

### 13.85.1.1 \_\_FFLASFFPACK\_fgemm\_winograd\_ip\_INL

```
#define __FFLASFFPACK_fgemm_winograd_ip_INL
```

## 13.86 fflas\_fgemv.inl File Reference

```
#include <givaro/zring.h>
```

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::Protected](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fgemv\\_INL](#)



## Functions

- `template<typename FloatElement, class Field >`  
`Field::Element_ptr fgemv_convert` (const `Field` &F, const `FFLAS_TRANSPOSE` ta, const `size_t` M, const `size_t` N, const `typename Field::Element` alpha, `typename Field::ConstElement_ptr` A, const `size_t` lda, `typename Field::ConstElement_ptr` X, const `size_t` incX, const `typename Field::Element` beta, `typename Field::Element_ptr` Y, const `size_t` incY)
- `template<class Field >`  
`Field::Element_ptr fgemv` (const `Field` &F, const `FFLAS_TRANSPOSE` ta, const `size_t` M, const `size_t` N, const `typename Field::Element` alpha, `typename Field::ConstElement_ptr` A, const `size_t` lda, `typename Field::ConstElement_ptr` X, const `size_t` incX, const `typename Field::Element` beta, `typename Field::Element_ptr` Y, const `size_t` incY, `MMHelper`< `Field`, `MMHelperAlgo::Classic`, `ModeCategories::ConvertTo`< `ElementCategories::MachineFloatTag` > > &H)
- `template<class Field >`  
`Field::Element_ptr fgemv` (const `Field` &F, const `FFLAS_TRANSPOSE` ta, const `size_t` M, const `size_t` N, const `typename Field::Element` alpha, `typename Field::ConstElement_ptr` A, const `size_t` lda, `typename Field::ConstElement_ptr` X, const `size_t` incX, const `typename Field::Element` beta, `typename Field::Element_ptr` Y, const `size_t` incY, `MMHelper`< `Field`, `MMHelperAlgo::Classic`, `ModeCategories::DelayedTag` > &H)
- `template<class Field >`  
`Field::Element_ptr fgemv` (const `Field` &F, const `FFLAS_TRANSPOSE` ta, const `size_t` M, const `size_t` N, const `typename Field::Element` alpha, `typename Field::ConstElement_ptr` A, const `size_t` lda, `typename Field::ConstElement_ptr` X, const `size_t` incX, const `typename Field::Element` beta, `typename Field::Element_ptr` Y, const `size_t` incY, `MMHelper`< `Field`, `MMHelperAlgo::Classic`, `ModeCategories::DefaultTag` > &H)
- `template<class Field >`  
`Field::Element_ptr fgemv` (const `Field` &F, const `FFLAS_TRANSPOSE` ta, const `size_t` M, const `size_t` N, const `typename Field::Element` alpha, `typename Field::ConstElement_ptr` A, const `size_t` lda, `typename Field::ConstElement_ptr` X, const `size_t` incX, const `typename Field::Element` beta, `typename Field::Element_ptr` Y, const `size_t` incY, `MMHelper`< `Field`, `MMHelperAlgo::Classic`, `ModeCategories::LazyTag` > &H)
- `template<class Field >`  
`Field::Element_ptr fgemv` (const `Field` &F, const `FFLAS_TRANSPOSE` TransA, const `size_t` M, const `size_t` N, const `typename Field::Element` alpha, `typename Field::ConstElement_ptr` A, const `size_t` lda, `typename Field::ConstElement_ptr` X, const `size_t` incX, const `typename Field::Element` beta, `typename Field::Element_ptr` Y, const `size_t` incY)  
*finite prime Field GEneral Matrix Vector multiplication.*
- `Givaro::ZRing< int64_t >::Element_ptr fgemv` (const `Givaro::ZRing< int64_t >` &F, const `FFLAS_TRANSPOSE` ta, const `size_t` M, const `size_t` N, const `int64_t` alpha, const `int64_t` \*A, const `size_t` lda, const `int64_t` \*X, const `size_t` incX, const `int64_t` beta, `int64_t` \*Y, const `size_t` incY, `MMHelper`< `Givaro::ZRing< int64_t >`, `MMHelperAlgo::Classic`, `ModeCategories::DefaultTag` > &H)
- `Givaro::DoubleDomain::Element_ptr fgemv` (const `Givaro::DoubleDomain` &F, const `FFLAS_TRANSPOSE` ta, const `size_t` M, const `size_t` N, const `Givaro::DoubleDomain::Element` alpha, const `Givaro::DoubleDomain::ConstElement_ptr` A, const `size_t` lda, const `Givaro::DoubleDomain::ConstElement_ptr` X, const `size_t` incX, const `Givaro::DoubleDomain::Element` beta, `Givaro::DoubleDomain::Element_ptr` Y, const `size_t` incY, `MMHelper`< `Givaro::DoubleDomain`, `MMHelperAlgo::Classic`, `ModeCategories::DefaultTag` > &H)
- `template<class Field >`  
`Field::Element_ptr fgemv` (const `Field` &F, const `FFLAS_TRANSPOSE` ta, const `size_t` M, const `size_t` N, const `typename Field::Element` alpha, const `typename Field::ConstElement_ptr` A, const `size_t` lda, const `typename Field::ConstElement_ptr` X, const `size_t` incX, const `typename Field::Element` beta, `typename Field::Element_ptr` Y, const `size_t` incY, `MMHelper`< `Field`, `MMHelperAlgo::Classic`, `ModeCategories::DefaultBoundedTag` > &H)
- `Givaro::FloatDomain::Element_ptr fgemv` (const `Givaro::FloatDomain` &F, const `FFLAS_TRANSPOSE` ta, const `size_t` M, const `size_t` N, const `Givaro::FloatDomain::Element` alpha, const `Givaro::FloatDomain::ConstElement_ptr` A, const `size_t` lda, const `Givaro::FloatDomain::ConstElement_ptr` X, const `size_t` incX, const `Givaro::FloatDomain::Element` beta, `Givaro::FloatDomain::Element_ptr` Y, const `size_t` incY, `MMHelper`< `Givaro::FloatDomain`, `MMHelperAlgo::Classic`, `ModeCategories::DefaultTag` > &H)



- `template<class Field , class Cut , class Param >`  
`Field::Element_ptr fgmv (const Field &F, const FFLAS_TRANSPOSE ta, const size_t m, const size_t n,`  
`const typename Field::Element alpha, const typename Field::ConstElement_ptr A, const size_t lda, const`  
`typename Field::ConstElement_ptr X, const size_t incX, const typename Field::Element beta, typename`  
`Field::Element_ptr Y, const size_t incY, ParSeqHelper::Parallel< Cut, Param > &parH)`
- `template<class Field >`  
`Field::Element_ptr fgmv (const Field &F, const FFLAS_TRANSPOSE ta, const size_t m, const size_t n,`  
`const typename Field::Element alpha, const typename Field::ConstElement_ptr A, const size_t lda, const`  
`typename Field::ConstElement_ptr X, const size_t incX, const typename Field::Element beta, typename`  
`Field::Element_ptr Y, const size_t incY, ParSeqHelper::Sequential &seqH)`

## 13.86.1 Macro Definition Documentation

### 13.86.1.1 \_\_FFLASFFPACK\_fgmv\_INL

```
#define __FFLASFFPACK_fgmv_INL
```

## 13.87 fflas\_fgmv\_mp.inl File Reference

```
#include "fflas-ffpack/field/rns-integer-mod.h"
```

### Namespaces

- namespace `FFLAS`

### Macros

- #define `__FFLASFFPACK_fgmv_mp_INL`

### Functions

- `FFPACK::rns_double::Element_ptr fgmv (const FFPACK::RNSInteger< FFPACK::rns_double > &F, const`  
`FFLAS_TRANSPOSE ta, const size_t M, const size_t N, const FFPACK::rns_double::Element alpha,`  
`FFPACK::rns_double::ConstElement_ptr A, const size_t lda, FFPACK::rns_double::ConstElement_ptr`  
`X, const size_t incX, const FFPACK::rns_double::Element beta, FFPACK::rns_double::Element_ptr Y,`  
`const size_t incY, MMHelper< FFPACK::RNSInteger< FFPACK::rns_double >, MMHelperAlgo::Classic,`  
`ModeCategories::DefaultTag > &H)`
- `FFPACK::rns_double::Element_ptr fgmv (const FFPACK::RNSIntegerMod< FFPACK::rns_double > &F,`  
`const FFLAS_TRANSPOSE ta, const size_t M, const size_t N, const FFPACK::rns_double::Element alpha,`  
`FFPACK::rns_double::ConstElement_ptr A, const size_t lda, FFPACK::rns_double::ConstElement_ptr X,`  
`const size_t incX, const FFPACK::rns_double::Element beta, FFPACK::rns_double::Element_ptr Y, const`  
`size_t incY, MMHelper< FFPACK::RNSIntegerMod< FFPACK::rns_double >, MMHelperAlgo::Classic,`  
`ModeCategories::DefaultTag > &H)`
- `Givaro::Integer * fgmv (const Givaro::ZRing< Givaro::Integer > &F, const FFLAS_TRANSPOSE`  
`ta, const size_t m, const size_t n, const Givaro::Integer alpha, Givaro::Integer *A, const size_t`  
`lda, Givaro::Integer *X, const size_t ldx, Givaro::Integer beta, Givaro::Integer *Y, const size_t`  
`ldy, MMHelper< Givaro::ZRing< Givaro::Integer >, MMHelperAlgo::Classic, ModeCategories::ConvertTo<`  
`ElementCategories::RNSElementTag > > &H)`
- `Givaro::Integer * fgmv (const Givaro::Modular< Givaro::Integer > &F, const FFLAS_TRANSPOSE`  
`ta, const size_t m, const size_t n, const Givaro::Integer alpha, Givaro::Integer *A, const size_t`  
`lda, Givaro::Integer *X, const size_t ldx, Givaro::Integer beta, Givaro::Integer *Y, const size_t`  
`ldy, MMHelper< Givaro::Modular< Givaro::Integer >, MMHelperAlgo::Classic, ModeCategories::ConvertTo<`  
`ElementCategories::RNSElementTag > > &H)`
- `template<size_t K1, size_t K2, class ParSeq >`  
`Reclnt::ruint< K1 > * fgmv (const Givaro::Modular< Reclnt::ruint< K1 >, Reclnt::ruint< K2 > >`  
`&F, const FFLAS_TRANSPOSE ta, const size_t m, const size_t n, const Reclnt::ruint< K1 > al-`  
`pha, const Reclnt::ruint< K1 > *A, const size_t lda, const Reclnt::ruint< K1 > *X, const size_t incx,`

```
Reclnt::ruint< K1 > beta, Reclnt::ruint< K1 > *Y, const size_t incy, MMHelper< Givaro::Modular<
Reclnt::ruint< K1 >, Reclnt::ruint< K2 > >, MMHelperAlgo::Classic, ModeCategories::ConvertTo<
ElementCategories::RNSElementTag >, ParSeq > &H)
```

## 13.87.1 Macro Definition Documentation

### 13.87.1.1 \_\_FFLASFFPACK\_fgmv\_mp\_INL

```
#define __FFLASFFPACK_fgmv_mp_INL
```

## 13.88 fflas\_fger.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::Protected](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fger\\_INL](#)

### Functions

- template<class [Field](#) >  
void [fger](#) (const [Field](#) &F, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) x, const size\_t incx, typename [Field::ConstElement\\_ptr](#) y, const size\_t incy, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
*fger: rank one update of a general matrix*
- template<class [FloatElement](#) , class [Field](#) >  
void [fger\\_convert](#) (const [Field](#) &F, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) x, const size\_t incx, typename [Field::ConstElement\\_ptr](#) y, const size\_t incy, typename [Field::Element\\_ptr](#) A, const size\_t lda)
- template<class [Field](#) >  
void [fger](#) (const [Field](#) &F, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) x, const size\_t incx, typename [Field::ConstElement\\_ptr](#) y, const size\_t incy, typename [Field::Element\\_ptr](#) A, const size\_t lda, [MMHelper](#)< [Field](#), [MMHelperAlgo::Classic](#), [ModeCategories::ConvertTo](#)< [ElementCategories::MachineFloatTag](#) > > &H)
- template<class [Field](#) , class [AnyTag](#) >  
void [fger](#) (const [Field](#) &F, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) x, const size\_t incx, typename [Field::ConstElement\\_ptr](#) y, const size\_t incy, typename [Field::Element\\_ptr](#) A, const size\_t lda, [MMHelper](#)< [Field](#), [MMHelperAlgo::Classic](#), [AnyTag](#) > &H)
- void [fger](#) (const [Givaro::DoubleDomain](#) &F, const size\_t M, const size\_t N, const [Givaro::DoubleDomain::Element](#) alpha, const [Givaro::DoubleDomain::ConstElement\\_ptr](#) x, const size\_t incx, const [Givaro::DoubleDomain::ConstElement\\_ptr](#) y, const size\_t incy, [Givaro::DoubleDomain::Element\\_ptr](#) A, const size\_t lda, [MMHelper](#)< [Givaro::DoubleDomain](#), [MMHelperAlgo::Classic](#), [ModeCategories::DefaultTag](#) > &H)
- template<class [Field](#) >  
void [fger](#) (const [Field](#) &F, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, const typename [Field::ConstElement\\_ptr](#) x, const size\_t incx, const typename [Field::ConstElement\\_ptr](#) y, const size\_t incy, typename [Field::Element\\_ptr](#) A, const size\_t lda, [MMHelper](#)< [Field](#), [MMHelperAlgo::Classic](#), [ModeCategories::DefaultBoundedTag](#) > &H)
- void [fger](#) (const [Givaro::FloatDomain](#) &F, const size\_t M, const size\_t N, const [Givaro::FloatDomain::Element](#) alpha, const [Givaro::FloatDomain::ConstElement\\_ptr](#) x, const size\_t incx, const [Givaro::FloatDomain::ConstElement\\_ptr](#) y, const size\_t incy, [Givaro::FloatDomain::Element\\_ptr](#) A, const size\_t lda, [MMHelper](#)< [Givaro::FloatDomain](#), [MMHelperAlgo::Classic](#), [ModeCategories::DefaultTag](#) > &H)
- template<class [Field](#) >  
void [fger](#) (const [Field](#) &F, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) x, const size\_t incx, typename [Field::ConstElement\\_ptr](#) y, const size\_t incy, typename [Field::Element\\_ptr](#) A, const size\_t lda, [MMHelper](#)< [Field](#), [MMHelperAlgo::Classic](#), [ModeCategories::LazyTag](#) > &H)

- `template<class Field >`  
`void fger (const Field &F, const size_t M, const size_t N, const typename Field::Element alpha, type-`  
`name Field::ConstElement_ptr x, const size_t incx, typename Field::ConstElement_ptr y, const size_t`  
`incy, typename Field::Element_ptr A, const size_t lda, MMHelper< Field, MMHelperAlgo::Classic,`  
`ModeCategories::DelayedTag > &H)`

## 13.88.1 Macro Definition Documentation

### 13.88.1.1 \_\_FFLASFFPACK\_fger\_INL

```
#define __FFLASFFPACK_fger_INL
```

## 13.89 fflas\_fger\_mp.inl File Reference

```
#include <givaro/modular-integer.h>
#include <givaro/zring.h>
#include "fflas-ffpack/fflas/fflas_helpers.inl"
#include "fflas-ffpack/fflas/fflas_fgemm/fgemm_classical_mp.inl"
#include "fflas-ffpack/field/rns-integer.h"
#include "fflas-ffpack/field/rns-integer-mod.h"
```

### Namespaces

- namespace [FFLAS](#)

### Macros

- `#define __FFPACK_fger_mp_INL`

### Functions

- `void fger (const Givaro::Modular< Givaro::Integer > &F, const size_t M, const size_t N, const typename`  
`Givaro::Integer alpha, typename Givaro::Integer *x, const size_t incx, typename Givaro::Integer *y, const`  
`size_t incy, typename Givaro::Integer *A, const size_t lda, MMHelper< Givaro::Modular< Givaro::Integer >,`  
`MMHelperAlgo::Classic, ModeCategories::ConvertTo< ElementCategories::RNSElementTag > > &H)`
- `template<typename RNS >`  
`void fger (const FFPACK::RNSInteger< RNS > &F, const size_t M, const size_t N, const typename`  
`FFPACK::RNSInteger< RNS >::Element alpha, typename FFPACK::RNSInteger< RNS >::Element_ptr x,`  
`const size_t incx, typename FFPACK::RNSInteger< RNS >::Element_ptr y, const size_t incy, typename`  
`FFPACK::RNSInteger< RNS >::Element_ptr A, const size_t lda, MMHelper< FFPACK::RNSInteger< RNS`  
`>, MMHelperAlgo::Classic, ModeCategories::DefaultTag > &H)`
- `template<typename RNS >`  
`void fger (const FFPACK::RNSIntegerMod< RNS > &F, const size_t M, const size_t N, const type-`  
`name FFPACK::RNSIntegerMod< RNS >::Element alpha, typename FFPACK::RNSIntegerMod< RNS`  
`>::Element_ptr x, const size_t incx, typename FFPACK::RNSIntegerMod< RNS >::Element_ptr y, const`  
`size_t incy, typename FFPACK::RNSIntegerMod< RNS >::Element_ptr A, const size_t lda, MMHelper<`  
`FFPACK::RNSIntegerMod< RNS >, MMHelperAlgo::Classic > &H)`

## 13.89.1 Macro Definition Documentation

### 13.89.1.1 \_\_FFPACK\_fger\_mp\_INL

```
#define __FFPACK_fger_mp_INL
```

## 13.90 fflas\_freduce.h File Reference

```
#include "fflas-ffpack/fflas/fflas_simd.h"
#include "fflas-ffpack/field/field-traits.h"
#include "fflas-ffpack/utils/cast.h"
#include "fflas-ffpack/fflas/fflas_freduce.inl"
```

### Data Structures

- struct [support\\_simd\\_mod< T >](#)
- struct [support\\_fast\\_mod< T >](#)
- struct [support\\_fast\\_mod< float >](#)
- struct [support\\_fast\\_mod< double >](#)
- struct [support\\_fast\\_mod< int64\\_t >](#)

### Namespaces

- namespace [FFLAS](#)

### Functions

- template<class [Field](#) >  
void [freduce](#) (const [Field](#) &F, const size\_t n, typename [Field::ConstElement\\_ptr](#) Y, const size\_t incY, typename [Field::Element\\_ptr](#) X, const size\_t incX)  
$$\text{freduce } x \leftarrow y \bmod F.$$
- template<class [Field](#) >  
void [freduce](#) (const [Field](#) &F, const size\_t n, typename [Field::Element\\_ptr](#) X, const size\_t incX)  
$$\text{freduce } x \leftarrow x \bmod F.$$
- template<class [Field](#) >  
void [freduce\\_constoverride](#) (const [Field](#) &F, const size\_t m, typename [Field::ConstElement\\_ptr](#) A, const size\_t incX)
- template<class [Field](#) , class ConstOtherElement\_ptr >  
void [finit](#) (const [Field](#) &F, const size\_t n, ConstOtherElement\_ptr Y, const size\_t incY, typename [Field::Element\\_ptr](#) X, const size\_t incX)
- template<class [Field](#) >  
void [finit](#) (const [Field](#) &F, const size\_t n, typename [Field::Element\\_ptr](#) X, const size\_t incX)  
$$\text{finit initializes } X \text{ in } F^{\$}.$$
- template<class [Field](#) >  
void [freduce](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
$$\text{freduce } A \leftarrow A \bmod F.$$
- template<class [Field](#) >  
void [freduce](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) uplo, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
$$\text{freduce for square symmetric matrices}$$
- template<class [Field](#) >  
void [pfreduce](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t numths)
- template<class [Field](#) >  
void [freduce](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
$$\text{freduce } A \leftarrow B \bmod F.$$
- template<class [Field](#) >  
void [freduce\\_constoverride](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda)

- template<class [Field](#) , class OtherElement\_ptr >  
void [finit](#) (const [Field](#) &F, const size\_t m, const size\_t n, const OtherElement\_ptr B, const size\_t ldb, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
 $finit\ A \leftarrow B mod F.$
- template<class [Field](#) >  
void [finit](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::Element\\_ptr](#) A, const size\_t lda)

## 13.91 fflas\_freduce.inl File Reference

```
#include <givaro/udl.h>
#include "fflas-ffpack/fflas/fflas_fassign.h"
```

### Data Structures

- struct [HelperMod](#)< [Field](#), [ElementCategories::MachineIntTag](#) >
- struct [HelperMod](#)< [Field](#), [FFLAS::ElementCategories::MachineFloatTag](#) >
- struct [HelperMod](#)< [Field](#), [FFLAS::ElementCategories::ArbitraryPrecIntTag](#) >
- struct [HelperMod](#)< [Field](#), [FFLAS::ElementCategories::FixedPrecIntTag](#) >

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::vectorised](#)
- namespace [FFLAS::vectorised::unswitch](#)
- namespace [FFLAS::details](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_freduce\\_INL](#)
- #define [FFLASFFPACK\\_COPY\\_REDUCE](#) 32 /\* TODO TO BENCHMARK LATER \*/

### Functions

- template<class T >  
std::enable\_if<!std::is\_integral< T >::value, T >::type [reduce](#) (T A, T B)
- template<class T >  
std::enable\_if< std::is\_integral< T >::value, T >::type [reduce](#) (T A, T B)
- template<> [Givaro::Integer reduce](#) ([Givaro::Integer](#) A, [Givaro::Integer](#) B)
- float [reduce](#) (float A, float B, float invB, float min, float max)
- double [reduce](#) (double A, double B, double invB, double min, double max)
- int64\_t [reduce](#) (int64\_t A, int64\_t p, double invp, double min, double max, int64\_t pow50rem)
- template<class [Field](#) >  
[Field::Element reduce](#) (typename [Field::Element](#) A, [HelperMod](#)< [Field](#), [ElementCategories::MachineIntTag](#) > &H)
- template<class [Field](#) >  
[Field::Element reduce](#) (typename [Field::Element](#) A, [HelperMod](#)< [Field](#), [ElementCategories::MachineFloatTag](#) > &H)
- template<class [Field](#) >  
[Field::Element reduce](#) (typename [Field::Element](#) A, [HelperMod](#)< [Field](#), [ElementCategories::ArbitraryPrecIntTag](#) > &H)
- template<class [Field](#) >  
std::enable\_if<![FFLAS::support\\_simd\\_mod](#)< typename [Field::Element](#) >::value &&[FFLAS::support\\_fast\\_mod](#)< typename [Field::Element](#) >::value, void >::type [modp](#) (const [Field](#) &F, typename [Field::ConstElement\\_ptr](#) U, const size\_t &n, typename [Field::Element\\_ptr](#) T, [HelperMod](#)< [Field](#) > &H)

- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type modp`  
`(const Field &F, typename Field::ConstElement_ptr U, const size_t &n, const size_t &incX, typename`  
`Field::Element_ptr T, HelperMod< Field > &H)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type modp` `(const`  
`Field &F, typename Field::ConstElement_ptr U, const size_t &n, typename Field::Element_ptr T)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type modp`  
`(const Field &F, typename Field::ConstElement_ptr U, const size_t &n, const size_t &incX, typename`  
`Field::Element_ptr T)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type freduce` `(const`  
`Field &F, const size_t m, typename Field::Element_ptr A, const size_t incX, FieldCategories::ModularTag)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type freduce`  
`(const Field &F, const size_t m, typename Field::ConstElement_ptr B, const size_t incY, typename`  
`Field::Element_ptr A, const size_t incX, FieldCategories::ModularTag)`
- `template<class Field, class FC >`  
`void freduce` `(const Field &F, const size_t m, typename Field::Element_ptr A, const size_t incX, FC)`
- `template<class Field, class FC >`  
`void freduce` `(const Field &F, const size_t m, typename Field::ConstElement_ptr B, const size_t incY, type-`  
`name Field::Element_ptr A, const size_t incX, FC)`

### 13.91.1 Macro Definition Documentation

#### 13.91.1.1 \_\_FFLASFFPACK\_fflas\_freduce\_INL

```
#define __FFLASFFPACK_fflas_freduce_INL
```

#### 13.91.1.2 FFLASFFPACK\_COPY\_REDUCE

```
#define FFLASFFPACK_COPY_REDUCE 32 /* TODO TO BENCHMARK LATER */
```

## 13.92 fflas\_freduce\_mp.inl File Reference

```
#include "fflas-ffpack/field/rns-integer-mod.h"
```

### Namespaces

- namespace [FFLAS](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_freduce\\_mp\\_INL](#)

### Functions

- `template<> void freduce` `(const FFPACK::RNSIntegerMod< FFPACK::rns_double > &F, const size_t n,`  
`FFPACK::RNSIntegerMod< FFPACK::rns_double >::Element_ptr A, size_t inc)`
- `template<> void freduce` `(const FFPACK::RNSIntegerMod< FFPACK::rns_double > &F, const size_t m,`  
`const size_t n, FFPACK::rns_double::Element_ptr A, size_t lda)`

### 13.92.1 Macro Definition Documentation

#### 13.92.1.1 \_\_FFLASFFPACK\_fflas\_freduce\_mp\_INL

```
#define __FFLASFFPACK_fflas_freduce_mp_INL
```

## 13.93 fflas\_freivalds.inl File Reference

### Namespaces

- namespace [FFLAS](#)

### Macros

- `#define __FFLASFFPACK_freivalds_INL`

### Functions

- `template<class Field >`  
`bool freivalds (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m,`  
`const size_t n, const size_t k, const typename Field::Element alpha, typename Field::ConstElement_ptr A,`  
`const size_t lda, typename Field::ConstElement_ptr B, const size_t ldb, typename Field::ConstElement_ptr`  
`C, const size_t ldc)`

*freivalds: Freivalds GEneral Matrix Multiply Random Check.*

### 13.93.1 Macro Definition Documentation

#### 13.93.1.1 \_\_FFLASFFPACK\_freivalds\_INL

```
#define __FFLASFFPACK_freivalds_INL
```

## 13.94 fflas\_fscal.h File Reference

```
#include "fflas_fscal.inl"
```

## 13.95 fflas\_fscal.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::vectorised](#)
- namespace [FFLAS::vectorised::unswitch](#)
- namespace [FFLAS::details](#)

### Macros

- `#define __FFLASFFPACK_fscal_INL`

### Functions

- `template<class Field >`  
`std::enable_if<!FFLAS::support_simd_mod< typenameField::Element >::value &&FFLAS::support_fast_mod<`  
`typenameField::Element >::value, void >::type scalp (const Field &F, typename Field::Element_ptr T, const`  
`typename Field::Element alpha, typename Field::ConstElement_ptr U, const size_t n, HelperMod< Field >`  
`&H)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type scalp`  
`(const Field &F, typename Field::Element_ptr T, const typename Field::Element alpha, typename`  
`Field::ConstElement_ptr U, const size_t n, const size_t &incX, HelperMod< Field > &H)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type scalp`  
`(const Field &F, typename Field::Element_ptr T, const typename Field::Element alpha, typename`  
`Field::ConstElement_ptr U, const size_t n, const size_t &incX, const size_t &incY, HelperMod< Field >`  
`&H)`

- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type scalp`  
`(const Field &F, typename Field::Element_ptr T, const typename Field::Element alpha, typename`  
`Field::ConstElement_ptr U, const size_t n)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type scalp`  
`(const Field &F, typename Field::Element_ptr T, const typename Field::Element alpha, typename`  
`Field::ConstElement_ptr U, const size_t n, const size_t &incX)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type scalp`  
`(const Field &F, typename Field::Element_ptr T, const typename Field::Element alpha, typename`  
`Field::ConstElement_ptr U, const size_t n, const size_t &incX, const size_t &incY)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type fscal`  
`(const Field &F, const size_t N, const typename Field::Element a, typename Field::Element_ptr X, const size_t incX,`  
`FieldCategories::ModularTag)`
- `template<class Field >`  
`std::enable_if< FFLAS::support_fast_mod< typenameField::Element >::value, void >::type fscal`  
`(const Field &F, const size_t N, const typename Field::Element a, typename Field::ConstElement_ptr X, const size_t incX,`  
`typename Field::Element_ptr Y, const size_t incY, FieldCategories::ModularTag)`
- `template<class Field, class FC >`  
`void fscal`  
`(const Field &F, const size_t n, const typename Field::Element a, typename Field::Element_ptr X,`  
`const size_t incX, FC)`
- `template<class Field, class FC >`  
`void fscal`  
`(const Field &F, const size_t N, const typename Field::Element a, typename Field::ConstElement_ptr`  
`X, const size_t incX, typename Field::Element_ptr Y, const size_t incY, FC)`
- `template<class Field >`  
`void fscal`  
`(const Field &F, const size_t n, const typename Field::Element alpha, typename Field::Element_ptr`  
`X, const size_t incX)`  

$$fscal\ x \leftarrow \alpha \cdot x.$$
- `template<class Field >`  
`void fscal`  
`(const Field &F, const size_t n, const typename Field::Element alpha, typename Field::ConstElement_ptr`  
`X, const size_t incX, typename Field::Element_ptr Y, const size_t incY)`  

$$fscal\ y \leftarrow \alpha \cdot x.$$
- `template<> void fscal`  
`(const Givaro::DoubleDomain &, const size_t N, const Givaro::DoubleDomain::Element a, Givaro::DoubleDomain::ConstElement_ptr x, const size_t incx, Givaro::DoubleDomain::Element_ptr y, const size_t incy)`
- `template<> void fscal`  
`(const Givaro::FloatDomain &, const size_t N, const Givaro::FloatDomain::Element a, Givaro::FloatDomain::ConstElement_ptr x, const size_t incx, Givaro::FloatDomain::Element_ptr y, const size_t incy)`
- `template<> void fscal`  
`(const Givaro::DoubleDomain &, const size_t N, const Givaro::DoubleDomain::Element a, Givaro::DoubleDomain::Element_ptr y, const size_t incy)`
- `template<> void fscal`  
`(const Givaro::FloatDomain &, const size_t N, const Givaro::FloatDomain::Element a, Givaro::FloatDomain::Element_ptr y, const size_t incy)`
- `template<class Field >`  
`void fscal`  
`(const Field &F, const size_t m, const size_t n, const typename Field::Element alpha, typename`  
`Field::Element_ptr A, const size_t lda)`  

$$fscal\ A \leftarrow a \cdot A.$$
- `template<class Field >`  
`void fscal`  
`(const Field &F, const size_t m, const size_t n, const typename Field::Element alpha, typename`  
`Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb)`  

$$fscal\ B \leftarrow a \cdot A.$$

## 13.95.1 Macro Definition Documentation

### 13.95.1.1 \_\_FFLASFFPACK\_fscal\_INL

```
#define __FFLASFFPACK_fscal_INL
```



## 13.96 fflas\_fscal\_mp.inl File Reference

```
#include "fflas-ffpack/field/rns-integer.h"
#include "fflas_fscal.h"
#include "fflas_fgemm.inl"
#include "fflas-ffpack/fflas/fflas_freduce_mp.inl"
```

### Namespaces

- namespace [FFLAS](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fscal\\_mp\\_INL](#)

### Functions

- template<> void [fscal](#) (const [FFPACK::RNSInteger](#)< [FFPACK::rns\\_double](#) > &F, const size\_t n, const [FFPACK::rns\\_double::Element](#) alpha, [FFPACK::rns\\_double::Element\\_ptr](#) A, const size\_t inc)
- template<> void [fscal](#) (const [FFPACK::RNSInteger](#)< [FFPACK::rns\\_double](#) > &F, const size\_t n, const [FFPACK::rns\\_double::Element](#) alpha, [FFPACK::rns\\_double::ConstElement\\_ptr](#) A, const size\_t Ainc, [FFPACK::rns\\_double::Element\\_ptr](#) B, const size\_t Binc)
- template<> void [fscal](#) (const [FFPACK::RNSInteger](#)< [FFPACK::rns\\_double](#) > &F, const size\_t m, const size\_t n, const [FFPACK::rns\\_double::Element](#) alpha, [FFPACK::rns\\_double::Element\\_ptr](#) A, const size\_t lda)
- template<> void [fscal](#) (const [FFPACK::RNSInteger](#)< [FFPACK::rns\\_double](#) > &F, const size\_t m, const size\_t n, const [FFPACK::rns\\_double::Element](#) alpha, [FFPACK::rns\\_double::ConstElement\\_ptr](#) A, const size\_t lda, [FFPACK::rns\\_double::Element\\_ptr](#) B, const size\_t ldb)
- template<> void [fscal](#) (const [FFPACK::RNSIntegerMod](#)< [FFPACK::rns\\_double](#) > &F, const size\_t n, const typename [FFPACK::RNSIntegerMod](#)< [FFPACK::rns\\_double](#) >::Element alpha, typename [FFPACK::RNSIntegerMod](#)< [FFPACK::rns\\_double](#) >::Element\_ptr A, const size\_t inc)
- template<> void [fscal](#) (const [FFPACK::RNSIntegerMod](#)< [FFPACK::rns\\_double](#) > &F, const size\_t n, const [FFPACK::rns\\_double::Element](#) alpha, [FFPACK::rns\\_double::ConstElement\\_ptr](#) A, const size\_t Ainc, [FFPACK::rns\\_double::Element\\_ptr](#) B, const size\_t Binc)
- template<> void [fscal](#) (const [FFPACK::RNSIntegerMod](#)< [FFPACK::rns\\_double](#) > &F, const size\_t m, const size\_t n, const [FFPACK::rns\\_double::Element](#) alpha, [FFPACK::rns\\_double::Element\\_ptr](#) A, const size\_t lda)
- template<> void [fscal](#) (const [FFPACK::RNSIntegerMod](#)< [FFPACK::rns\\_double](#) > &F, const size\_t m, const size\_t n, const [FFPACK::rns\\_double::Element](#) alpha, [FFPACK::rns\\_double::ConstElement\\_ptr](#) A, const size\_t lda, [FFPACK::rns\\_double::Element\\_ptr](#) B, const size\_t ldb)

### 13.96.1 Macro Definition Documentation

#### 13.96.1.1 \_\_FFLASFFPACK\_fscal\_mp\_INL

```
#define __FFLASFFPACK_fscal_mp_INL
```

## 13.97 fflas\_fsyr2k.inl File Reference

```
#include <fflas-ffpack/utils/fflas_io.h>
```

### Namespaces

- namespace [FFLAS](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_fsyr2k\\_INL](#)

## Functions

- `template<class Field >`  
`Field::Element_ptr fsyr2k` (const `Field` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` n, const `size_t` k, const typename `Field::Element` alpha, typename `Field::ConstElement_ptr` A, const `size_t` lda, typename `Field::ConstElement_ptr` B, const `size_t` ldb, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc)  
*fsyr2k: Symmetric Rank 2K update*

## 13.97.1 Macro Definition Documentation

### 13.97.1.1 \_\_FFLASFFPACK\_fflas\_fsyr2k\_INL

```
#define __FFLASFFPACK_fflas_fsyr2k_INL
```

## 13.98 fflas\_fsyrk.inl File Reference

### Namespaces

- namespace `FFLAS`
- namespace `FFLAS::Protected`

### Macros

- `#define __FFLASFFPACK_fflas_fsyrk_INL`

### Functions

- `template<class NewField , class Field , class FieldMode >`  
`Field::Element_ptr fsyrk_convert` (const `Field` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` N, const `size_t` K, const typename `Field::Element` alpha, typename `Field::ConstElement_ptr` A, const `size_t` lda, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc, `MMHelper`< `Field`, `MMHelperAlgo::Classic`, `FieldMode` > &H)
- `template<class Field >`  
`Field::Element_ptr fsyrk` (const `Field` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` n, const `size_t` k, const typename `Field::Element` alpha, typename `Field::ConstElement_ptr` A, const `size_t` lda, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc)  
*fsyrk: Symmetric Rank K update*
- `template<class Field >`  
`Field::Element_ptr fsyrk` (const `Field` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` N, const `size_t` K, const typename `Field::Element` alpha, typename `Field::ConstElement_ptr` A, const `size_t` lda, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc, const `ParSeqHelper::Sequential` seq)
- `template<class Field >`  
`Field::Element_ptr fsyrk` (const `Field` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` N, const `size_t` K, const typename `Field::Element` alpha, typename `Field::ConstElement_ptr` A, const `size_t` lda, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc, `MMHelper`< `Field`, `MMHelperAlgo::Classic`, `ModeCategories::DefaultTag` > &H)
- `template<class Field >`  
`Field::Element_ptr fsyrk` (const `Field` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` N, const `size_t` K, const typename `Field::Element` alpha, typename `Field::ConstElement_ptr` A, const `size_t` lda, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc, `MMHelper`< `Field`, `MMHelperAlgo::Classic`, `ModeCategories::ConvertTo`< `ElementCategories::MachineFloatTag` >, `ParSeqHelper::Sequential` > &H)
- `template<class Field , class AlgoT , class ParSeqTrait >`  
`void ScalAndReduce` (const `Field` &F, const `FFLAS_UPLO` UpLo, const `size_t` N, const typename `Field::Element` alpha, typename `Field::Element_ptr` A, const `size_t` lda, const `MMHelper`< `Field`, `AlgoT`, `ModeCategories::LazyTag`, `ParSeqTrait` > &H)

- `template<class Field >`  
`Field::Element_ptr fsyrk` (const `Field` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` N, const `size_t` K, const typename `Field::Element` alpha, typename `Field::ConstElement_ptr` A, const `size_t` lda, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc, `MMHelper`< `Field`, `MMHelperAlgo::Classic`, `ModeCategories::DelayedTag` > &H)
- `template<class Field >`  
`Field::Element_ptr fsyrk` (const `Field` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` N, const `size_t` K, const typename `Field::Element` alpha, typename `Field::ConstElement_ptr` A, const `size_t` lda, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc, `MMHelper`< `Field`, `MMHelperAlgo::Classic`, `ModeCategories::LazyTag` > &H)
- `template<class Field , typename Mode >`  
`Field::Element_ptr fsyrk` (const `Field` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` N, const `size_t` K, const typename `Field::Element` alpha, typename `Field::ConstElement_ptr` A, const `size_t` lda, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc, `MMHelper`< `Field`, `MMHelperAlgo::DivideAndConquer`, `Mode` > &H)
- `template<class Field >`  
`Field::Element_ptr fsyrk` (const `Field` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` N, const `size_t` K, const typename `Field::Element` alpha, typename `Field::ConstElement_ptr` A, const `size_t` lda, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc, `MMHelper`< `Field`, `MMHelperAlgo::Classic`, `ModeCategories::DefaultBoundedTag` > &H)
- `Givaro::FloatDomain::Element_ptr fsyrk` (const `Givaro::FloatDomain` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` N, const `size_t` K, const `Givaro::FloatDomain::Element` alpha, `Givaro::FloatDomain::ConstElement_ptr` A, const `size_t` lda, const `Givaro::FloatDomain::Element` beta, `Givaro::FloatDomain::Element_ptr` C, const `size_t` ldc, `MMHelper`< `Givaro::FloatDomain`, `MMHelperAlgo::Classic`, `ModeCategories::DefaultTag` > &H)
- `Givaro::DoubleDomain::Element_ptr fsyrk` (const `Givaro::DoubleDomain` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` N, const `size_t` K, const `Givaro::DoubleDomain::Element` alpha, `Givaro::DoubleDomain::ConstElement_ptr` A, const `size_t` lda, const `Givaro::DoubleDomain::Element` beta, `Givaro::DoubleDomain::Element_ptr` C, const `size_t` ldc, `MMHelper`< `Givaro::DoubleDomain`, `MMHelperAlgo::Classic`, `ModeCategories::DefaultTag` > &H)
- `template<class Field >`  
`Field::Element_ptr fsyrk` (const `Field` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` n, const `size_t` k, const typename `Field::Element` alpha, typename `Field::Element_ptr` A, const `size_t` lda, typename `Field::ConstElement_ptr` D, const `size_t` incD, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc, const `size_t` threshold=`__FFLASFFPACK_FSYRK_THRESHOLD`)  
*fsyrk: Symmetric Rank K update with diagonal scaling*
- `template<class Field >`  
`Field::Element_ptr fsyrk` (const `Field` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` N, const `size_t` K, const typename `Field::Element` alpha, typename `Field::Element_ptr` A, const `size_t` lda, typename `Field::ConstElement_ptr` D, const `size_t` incD, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc, const `ParSeqHelper::Sequential` seq, const `size_t` threshold)
- `template<class Field , class Cut , class Param >`  
`Field::Element_ptr fsyrk` (const `Field` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` N, const `size_t` K, const typename `Field::Element` alpha, typename `Field::Element_ptr` A, const `size_t` lda, typename `Field::ConstElement_ptr` D, const `size_t` incD, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc, const `ParSeqHelper::Parallel`< `Cut`, `Param` > par, const `size_t` threshold)
- `template<class Field >`  
`Field::Element_ptr fsyrk` (const `Field` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` n, const `size_t` k, const typename `Field::Element` alpha, typename `Field::Element_ptr` A, const `size_t` lda, typename `Field::ConstElement_ptr` D, const `size_t` incD, const `std::vector< bool >` &twoBlock, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc, const `size_t` threshold=`__FFLASFFPACK_FSYRK_THRESHOLD`)  
*fsyrk: Symmetric Rank K update with diagonal scaling*

## 13.98.1 Macro Definition Documentation

### 13.98.1.1 \_\_FFLASFFPACK\_fflas\_fsyrr\_INL

```
#define __FFLASFFPACK_fflas_fsyrr_INL
```

## 13.99 fflas\_fsyrr\_strassen.inl File Reference

```
#include <givaro/givintsqrootmod.h>
```

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::Protected](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_fsyrr\\_strassen\\_INL](#)

### Functions

- template<class [Field](#) , class Element , class AlgoT , class ParSeqTrait >  
bool [NeedPreScalReduction](#) (Element &Outmin, Element &Outmax, Element &Op1min, Element &Op1max, const Element &x, [MMHelper](#)< [Field](#), AlgoT, [ModeCategories::LazyTag](#), ParSeqTrait > &WH)
- template<class [Field](#) , class Element , class AlgoT , class ModeT , class ParSeqTrait >  
bool [NeedPreScalReduction](#) (Element &Outmin, Element &Outmax, Element &Op1min, Element &Op1max, const Element &x, [MMHelper](#)< [Field](#), AlgoT, ModeT, ParSeqTrait > &WH)
- template<class [Field](#) , class Element , class AlgoT , class ParSeqTrait >  
bool [NeedPreApyReduction](#) (Element &Outmin, Element &Outmax, Element &Op1min, Element &Op1max, Element &Op2min, Element &Op2max, const Element &x, [MMHelper](#)< [Field](#), AlgoT, [ModeCategories::LazyTag](#), ParSeqTrait > &WH)
- template<class [Field](#) , class Element , class AlgoT , class ModeT , class ParSeqTrait >  
bool [NeedPreApyReduction](#) (Element &Outmin, Element &Outmax, Element &Op1min, Element &Op1max, Element &Op2min, Element &Op2max, const Element &x, [MMHelper](#)< [Field](#), AlgoT, ModeT, ParSeqTrait > &WH)
- template<class [Field](#) , class FieldTrait >  
void [computeS1S2](#) (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) trans, const size\_t N, const size\_t K, const typename [Field::Element](#) x, const typename [Field::Element](#) y, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) S, const size\_t lds, typename [Field::Element\\_ptr](#) T, const size\_t ldt, [MMHelper](#)< [Field](#), [MMHelperAlgo::Winograd](#), FieldTrait > &WH)
- template<class [Field](#) >  
[Field::Element\\_ptr](#) [fsyrr](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) UpLo, const [FFLAS\\_TRANSPOSE](#) trans, const size\_t N, const size\_t K, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc, [MMHelper](#)< [Field](#), [MMHelperAlgo::Winograd](#), [ModeCategories::DelayedTag](#), ParSeqHelper::Sequential > &H)
- template<class [Field](#) , class Mode >  
[Field::Element\\_ptr](#) [fsyrr](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) UpLo, const [FFLAS\\_TRANSPOSE](#) trans, const size\_t N, const size\_t K, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc, [MMHelper](#)< [Field](#), [MMHelperAlgo::Winograd](#), Mode > &H)
- template<class [Field](#) , class FieldTrait >  
[Field::Element\\_ptr](#) [fsyrr\\_strassen](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) uplo, const [FFLAS\\_TRANSPOSE](#) trans, const size\_t N, const size\_t K, const typename [Field::Element](#) y1, const typename [Field::Element](#) y2, const typename [Field::Element](#) alpha, typename [Field::Element\\_ptr](#) A, const size\_t lda, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc, [MMHelper](#)< [Field](#), [MMHelperAlgo::Winograd](#), FieldTrait > &WH)

### 13.99.1 Macro Definition Documentation

#### 13.99.1.1 \_\_FFLASFFPACK\_fflas\_fsyrrk\_strassen\_INL

```
#define __FFLASFFPACK_fflas_fsyrrk_strassen_INL
```

## 13.100 fflas\_ftrmm.inl File Reference

### Namespaces

- namespace [FFLAS](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_ftrmm\\_INL](#)

### Functions

- template<class [Field](#) >  
void [ftrmm](#) (const [Field](#) &F, const [FFLAS\\_SIDE](#) Side, const [FFLAS\\_UPLO](#) Uplo, const [FFLAS\\_TRANSPOSE](#) TransA, const [FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) B, const size\_t ldb)  
*ftrmm: **TR**angular **M**atrix **M**ultiply.*
- template<class [Field](#) >  
void [ftrmm](#) (const [Field](#) &F, const [FFLAS\\_SIDE](#) Side, const [FFLAS\\_UPLO](#) Uplo, const [FFLAS\\_TRANSPOSE](#) TransA, const [FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc)  
*ftrmm: **TR**angular **M**atrix **M**ultiply with 3 operands Computes  $C \leftarrow \alpha \text{op}(A)B + \text{beta}C$  or  $C \leftarrow \alpha \text{Bop}(A) + \text{beta}C$ .*

### 13.100.1 Macro Definition Documentation

#### 13.100.1.1 \_\_FFLASFFPACK\_ftrmm\_INL

```
#define __FFLASFFPACK_ftrmm_INL
```

## 13.101 fflas\_ftrsm.inl File Reference

### Namespaces

- namespace [FFLAS](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_ftrsm\\_INL](#)

### Functions

- template<class [Field](#) >  
void [ftrsm](#) (const [Field](#) &F, const [FFLAS\\_SIDE](#) Side, const [FFLAS\\_UPLO](#) Uplo, const [FFLAS\\_TRANSPOSE](#) TransA, const [FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) B, const size\_t ldb)
- template<class [Field](#) >  
void [ftrsm](#) (const [Field](#) &F, const [FFLAS\\_SIDE](#) Side, const [FFLAS\\_UPLO](#) Uplo, const [FFLAS\\_TRANSPOSE](#) TransA, const [FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) B, const size\_t ldb, const [ParSeqHelper::Sequential](#) &PSH)

- `template<class Field , class Cut , class Param >`  
`void ftrsm (const Field &F, const FFLAS_SIDE Side, const FFLAS_UPLO Uplo, const FFLAS_TRANSPOSE`  
`TransA, const FFLAS_DIAG Diag, const size_t M, const size_t N, const typename Field::Element alpha,`  
`typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb, const`  
`ParSeqHelper::Parallel< Cut, Param > &PSH)`
- `template<class Field , class ParSeqTrait = ParSeqHelper::Sequential>`  
`void ftrsm (const Field &F, const FFLAS_SIDE Side, const FFLAS_UPLO Uplo, const FFLAS_TRANSPOSE`  
`TransA, const FFLAS_DIAG Diag, const size_t M, const size_t N, const typename Field::Element alpha, type-`  
`name Field::Element_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb, TRSMHelper<`  
`StructureHelper::Recursive, ParSeqTrait > &H)`

### 13.101.1 Macro Definition Documentation

#### 13.101.1.1 \_\_FFLASFFPACK\_ftrsm\_INL

```
#define __FFLASFFPACK_ftrsm_INL
```

## 13.102 fflas\_ftrsm\_mp.inl File Reference

triangular system with matrix right hand side over multiprecision domain (either over Z or over Z/pZ)

```
#include <cmath>
#include <givaro/modular-integer.h>
#include <givaro/givinteger.h>
#include "fflas-ffpack/fflas/fflas_bounds.inl"
#include "fflas-ffpack/fflas/fflas_level3.inl"
#include "fflas-ffpack/field/rns-integer-mod.h"
#include "fflas-ffpack/field/rns-integer.h"
```

### Namespaces

- namespace [FFLAS](#)

### Macros

- `#define __FFPACK_ftrsm_mp_INL`

### Functions

- `void ftrsm (const Givaro::Modular< Givaro::Integer > &F, const FFLAS_SIDE Side, const FFLAS_UPLO`  
`Uplo, const FFLAS_TRANSPOSE TransA, const FFLAS_DIAG Diag, const size_t M, const size_t N, const`  
`Givaro::Integer alpha, const Givaro::Integer *A, const size_t lda, Givaro::Integer *B, const size_t ldb)`
- `void cblas_impftrsm (const enum FFLAS_ORDER Order, const enum FFLAS_SIDE Side, const enum`  
`FFLAS_UPLO Uplo, const enum FFLAS_TRANSPOSE TransA, const enum FFLAS_DIAG Diag, const`  
`int M, const int N, const FFPACK::rns_double_elt alpha, FFPACK::rns_double_elt_cstptr A, const int lda,`  
`FFPACK::rns_double_elt_ptr B, const int ldb)`

### 13.102.1 Detailed Description

triangular system with matrix right hand side over multiprecision domain (either over Z or over Z/pZ)

### 13.102.2 Macro Definition Documentation

#### 13.102.2.1 \_\_FFPACK\_ftrsm\_mp\_INL

```
#define __FFPACK_ftrsm_mp_INL
```

## 13.103 fflas\_ftrsv.inl File Reference

### Namespaces

- namespace [FFLAS](#)

### Macros

- `#define __FFLASFFPACK_ftrsv_INL`

### Functions

- `template<class Field >`  
`void ftrsv (const Field &F, const FFLAS\_UPLO Uplo, const FFLAS\_TRANSPOSE TransA, const FFLAS\_DIAG Diag, const size_t N, typename Field::ConstElement\_ptr A, const size_t lda, typename Field::Element\_ptr X, int incX)`  
*ftrsv: TRIangular System solve with Vector Computes  $X \leftarrow \text{op}(A^{-1})X$*

### 13.103.1 Macro Definition Documentation

#### 13.103.1.1 \_\_FFLASFFPACK\_ftrsv\_INL

```
#define __FFLASFFPACK_ftrsv_INL
```

## 13.104 fflas\_helpers.inl File Reference

```
#include "fflas-ffpack/field/field-traits.h"
#include "fflas-ffpack/paladin/parallel.h"
#include "fflas-ffpack/utils/flimits.h"
#include <algorithm>
```

### Data Structures

- struct [Auto](#)
- struct [Classic](#)
- struct [DivideAndConquer](#)
- struct [Winograd](#)
- struct [WinogradPar](#)
- struct [Bini](#)
- struct [AlgoChooser](#)< [ModeT](#), [ParSeq](#) >
- struct [AlgoChooser](#)< [ModeCategories::ConvertTo](#)< [ElementCategories::RNSElementTag](#) >, [ParSeq](#) >
- struct [MMHelper](#)< [Field](#), [AlgoTrait](#), [ModeCategories::DefaultTag](#), [ParSeqTrait](#) >  
*FGEMM Helper for Default and ConvertTo modes of operation.*
- struct [MMHelper](#)< [Field](#), [AlgoTrait](#), [ModeCategories::ConvertTo](#)< [Dest](#) >, [ParSeqTrait](#) >
- struct [MMHelper](#)< [Field](#), [AlgoTrait](#), [ModeTrait](#), [ParSeqTrait](#) >
- struct [Recursive](#)
- struct [Iterative](#)
- struct [Hybrid](#)
- struct [TRSMHelper](#)< [ReclterTrait](#), [ParSeqTrait](#) >  
*TRSM Helper.*

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::Protected](#)
- namespace [FFLAS::MMHelperAlgo](#)
- namespace [FFLAS::StructureHelper](#)  
*StructureHelper for ftrsm.*

## Macros

- `#define __FFLASFFPACK_fflas_fflas_mmhelper_INL`

## Functions

- `template<class Field >`  
`int WinogradSteps (const Field &F, const size_t &m)`  
*Computes the number of recursive levels to perform.*
- `template<class DFE >`  
`size_t min_types (const DFE &k)`
- `template<> size_t min_types (const Reclnt::rint< 6 > &k)`
- `template<> size_t min_types (const Reclnt::rint< 7 > &k)`
- `template<> size_t min_types (const Reclnt::rint< 8 > &k)`
- `template<> size_t min_types (const Reclnt::rint< 9 > &k)`
- `template<> size_t min_types (const Reclnt::rint< 10 > &k)`
- `template<> size_t min_types (const Givaro::Integer &k)`
- `template<class T >`  
`bool unfit (T x)`
- `template<> bool unfit (int64_t x)`
- `template<size_t K>`  
`bool unfit (Reclnt::rint< K > x)`
- `template<> bool unfit (Reclnt::rint< 6 > x)`

### 13.104.1 Macro Definition Documentation

#### 13.104.1.1 \_\_FFLASFFPACK\_fflas\_fflas\_mmhelper\_INL

```
#define __FFLASFFPACK_fflas_fflas_mmhelper_INL
```

## 13.105 igemm.doxy File Reference

## 13.106 igemm.h File Reference

```
#include "igemm_kernels.h"
#include "igemm_tools.h"
#include "fflas-ffpack/utils/fflas_memory.h"
#include "igemm.inl"
```

## Namespaces

- namespace `FFLAS`
- namespace `FFLAS::Protected`

## Enumerations

- enum `number_kind` { `zero` =0 , `one` =1 , `mone` =-1 , `other` =2 }

## Functions

- `template<enum FFLAS_TRANSPOSE tA, enum FFLAS_TRANSPOSE tB>`  
`void igemm_colmajor (size_t rows, size_t cols, size_t depth, const int64_t alpha, const int64_t *A, size_t lda, const int64_t *B, size_t ldb, int64_t *C, size_t ldc)`
- `template<enum FFLAS_TRANSPOSE tA, enum FFLAS_TRANSPOSE tB, enum number_kind alpha_kind>`  
`void igemm_colmajor (size_t rows, size_t cols, size_t depth, const int64_t alpha, const int64_t *A, size_t lda, const int64_t *B, size_t ldb, int64_t *C, size_t ldc)`



- void [igemm](#) (const enum [FFLAS\\_TRANSPOSE](#) TransA, const enum [FFLAS\\_TRANSPOSE](#) TransB, size\_t rows, size\_t cols, size\_t depth, const int64\_t alpha, const int64\_t \*A, size\_t lda, const int64\_t \*B, size\_t ldb, const int64\_t beta, int64\_t \*C, size\_t ldc)
- void [igemm\\_](#) (const enum [FFLAS\\_ORDER](#) Order, const enum [FFLAS\\_TRANSPOSE](#) TransA, const enum [FFLAS\\_TRANSPOSE](#) TransB, const size\_t M, const size\_t N, const size\_t K, const int64\_t alpha, const int64\_t \*A, const size\_t lda, const int64\_t \*B, const size\_t ldb, const int64\_t beta, int64\_t \*C, const size\_t ldc)

## 13.107 igemm.inl File Reference

```
#include "fflas-ffpack/utils/fflas_memory.h"
```

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::Protected](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_igemm\\_igemm\\_INL](#)

### Functions

- template<enum [FFLAS\\_TRANSPOSE](#) tA, enum [FFLAS\\_TRANSPOSE](#) tB>  
void [igemm\\_colmajor](#) (size\_t rows, size\_t cols, size\_t depth, const int64\_t alpha, const int64\_t \*A, size\_t lda, const int64\_t \*B, size\_t ldb, int64\_t \*C, size\_t ldc)
- template<enum [FFLAS\\_TRANSPOSE](#) tA, enum [FFLAS\\_TRANSPOSE](#) tB, enum [number\\_kind](#) alpha\_kind>  
void [igemm\\_colmajor](#) (size\_t rows, size\_t cols, size\_t depth, const int64\_t alpha, const int64\_t \*A, size\_t lda, const int64\_t \*B, size\_t ldb, int64\_t \*C, size\_t ldc)
- void [igemm](#) (const enum [FFLAS\\_TRANSPOSE](#) TransA, const enum [FFLAS\\_TRANSPOSE](#) TransB, size\_t rows, size\_t cols, size\_t depth, const int64\_t alpha, const int64\_t \*A, size\_t lda, const int64\_t \*B, size\_t ldb, const int64\_t beta, int64\_t \*C, size\_t ldc)
- void [igemm\\_](#) (const enum [FFLAS\\_ORDER](#) Order, const enum [FFLAS\\_TRANSPOSE](#) TransA, const enum [FFLAS\\_TRANSPOSE](#) TransB, const size\_t M, const size\_t N, const size\_t K, const int64\_t alpha, const int64\_t \*A, const size\_t lda, const int64\_t \*B, const size\_t ldb, const int64\_t beta, int64\_t \*C, const size\_t ldc)

### 13.107.1 Macro Definition Documentation

#### 13.107.1.1 [\\_\\_FFLASFFPACK\\_fflas\\_igemm\\_igemm\\_INL](#)

```
#define __FFLASFFPACK_fflas_igemm_igemm_INL
```

## 13.108 igemm\_kernels.h File Reference

```
#include "igemm_kernels.inl"
```

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::details](#)

### Functions

- template<enum [number\\_kind](#) K>  
void [igebb44](#) (size\_t i, size\_t j, size\_t depth, size\_t pdeth, const int64\_t alpha, const int64\_t \*bIA, const int64\_t \*bIB, int64\_t \*C, size\_t ldc)

- `template<enum number\_kind K>`  
`void igebb24 (size_t i, size_t j, size_t depth, size_t pdeth, const int64_t alpha, const int64_t *blA, const int64_t *blB, int64_t *C, size_t ldc)`
- `template<enum number\_kind K>`  
`void igebb14 (size_t i, size_t j, size_t depth, size_t pdeth, const int64_t alpha, const int64_t *blA, const int64_t *blB, int64_t *C, size_t ldc)`
- `template<enum number\_kind K>`  
`void igebb41 (size_t i, size_t j, size_t depth, size_t pdeth, const int64_t alpha, const int64_t *blA, const int64_t *blB, int64_t *C, size_t ldc)`
- `template<enum number\_kind K>`  
`void igebb21 (size_t i, size_t j, size_t depth, size_t pdeth, const int64_t alpha, const int64_t *blA, const int64_t *blB, int64_t *C, size_t ldc)`
- `template<enum number\_kind K>`  
`void igebb11 (size_t i, size_t j, size_t depth, size_t pdeth, const int64_t alpha, const int64_t *blA, const int64_t *blB, int64_t *C, size_t ldc)`
- `template<enum number\_kind K>`  
`void igebp (size_t rows, size_t cols, size_t depth, const int64_t alpha, const int64_t *blockA, size_t lda, const int64_t *blockB, size_t ldb, int64_t *C, size_t ldc)`

## 13.109 `igemm_kernels.inl` File Reference

```
#include "fflas-ffpack/utils/fflas_memory.h"
#include "igemm_tools.h"
```

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::details](#)

### Macros

- `#define \_\_FFLASFFPACK\_fflas\_igemm\_igemm\_kernels\_INL`

### Functions

- `template<enum number\_kind K>`  
`void igebb44 (size_t i, size_t j, size_t depth, size_t pdeth, const int64_t alpha, const int64_t *blA, const int64_t *blB, int64_t *C, size_t ldc)`
- `template<enum number\_kind K>`  
`void igebb24 (size_t i, size_t j, size_t depth, size_t pdeth, const int64_t alpha, const int64_t *blA, const int64_t *blB, int64_t *C, size_t ldc)`
- `template<enum number\_kind K>`  
`void igebb14 (size_t i, size_t j, size_t depth, size_t pdeth, const int64_t alpha, const int64_t *blA, const int64_t *blB, int64_t *C, size_t ldc)`
- `template<enum number\_kind K>`  
`void igebb41 (size_t i, size_t j, size_t depth, size_t pdeth, const int64_t alpha, const int64_t *blA, const int64_t *blB, int64_t *C, size_t ldc)`
- `template<enum number\_kind K>`  
`void igebb21 (size_t i, size_t j, size_t depth, size_t pdeth, const int64_t alpha, const int64_t *blA, const int64_t *blB, int64_t *C, size_t ldc)`
- `template<enum number\_kind K>`  
`void igebb11 (size_t i, size_t j, size_t depth, size_t pdeth, const int64_t alpha, const int64_t *blA, const int64_t *blB, int64_t *C, size_t ldc)`
- `template<enum number\_kind K>`  
`void igebp (size_t rows, size_t cols, size_t depth, const int64_t alpha, const int64_t *blockA, size_t lda, const int64_t *blockB, size_t ldb, int64_t *C, size_t ldc)`

### 13.109.1 Macro Definition Documentation

#### 13.109.1.1 \_\_FFLASFFPACK\_fflas\_igemm\_igemm\_kernels\_INL

```
#define __FFLASFFPACK_fflas_igemm_igemm_kernels_INL
```

## 13.110 igemm\_tools.h File Reference

```
#include "igemm_tools.inl"
```

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::details](#)

### Functions

- template<size\_t k, bool transpose>  
void [pack\\_lhs](#) (int64\_t \*XX, const int64\_t \*X, size\_t ldx, size\_t rows, size\_t cols)
- template<size\_t k, bool transpose>  
void [pack\\_rhs](#) (int64\_t \*XX, const int64\_t \*X, size\_t ldx, size\_t rows, size\_t cols)
- void [gebp](#) (size\_t rows, size\_t cols, size\_t depth, int64\_t \*C, size\_t ldc, const int64\_t \*blockA, size\_t lda, const int64\_t \*BlockB, size\_t ldb, int64\_t \*BlockW)
- void [BlockingFactor](#) (size\_t &m, size\_t &n, size\_t &k)

## 13.111 igemm\_tools.inl File Reference

```
#include "fflas-ffpack/fflas/fflas_simd.h"
```

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::details](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_igemm\\_igemm\\_tools\\_INL](#)

### Functions

- template<size\_t k, bool transpose>  
void [pack\\_rhs](#) (int64\_t \*XX, const int64\_t \*X, size\_t ldx, size\_t rows, size\_t cols)
- template<size\_t k, bool transpose>  
void [pack\\_lhs](#) (int64\_t \*XX, const int64\_t \*X, size\_t ldx, size\_t rows, size\_t cols)
- void [BlockingFactor](#) (size\_t &m, size\_t &n, size\_t &k)

### 13.111.1 Macro Definition Documentation

#### 13.111.1.1 \_\_FFLASFFPACK\_fflas\_igemm\_igemm\_tools\_INL

```
#define __FFLASFFPACK_fflas_igemm_igemm_tools_INL
```

## 13.112 fflas\_level1.inl File Reference

### Namespaces

- namespace [FFLAS](#)

## Macros

- `#define __FFLASFFPACK_fflas_fflas_level1_INL`

## Functions

- `template<class Field >`  
`void freduce (const Field &F, const size_t n, typename Field::Element_ptr X, const size_t incX)`  

$$freduce\ x \leftarrow x \bmod F.$$
- `template<class Field >`  
`void freduce (const Field &F, const size_t n, typename Field::ConstElement_ptr Y, const size_t incY, typename Field::Element_ptr X, const size_t incX)`  

$$freduce\ x \leftarrow y \bmod F.$$
- `template<class Field, class OtherElement_ptr >`  
`void finit (const Field &F, const size_t n, const OtherElement_ptr Y, const size_t incY, typename Field::Element_ptr X, const size_t incX)`  

$$finit\ x \leftarrow y \bmod F.$$
- `template<class Field >`  
`void finit (const Field &F, const size_t n, typename Field::Element_ptr X, const size_t incX)`  

$$finit\ \text{Initializes } X \text{ in } F.$$
- `template<class Field, class OtherElement_ptr >`  
`void fconvert (const Field &F, const size_t n, OtherElement_ptr X, const size_t incX, typename Field::ConstElement_ptr Y, const size_t incY)`  

$$fconvert\ x \leftarrow y \bmod F.$$
- `template<class Field >`  
`void fnegin (const Field &F, const size_t n, typename Field::Element_ptr X, const size_t incX)`  

$$fnegin\ x \leftarrow -x.$$
- `template<class Field >`  
`void fneg (const Field &F, const size_t n, typename Field::ConstElement_ptr Y, const size_t incY, typename Field::Element_ptr X, const size_t incX)`  

$$fneg\ x \leftarrow -y.$$
- `template<class Field >`  
`void fzero (const Field &F, const size_t n, typename Field::Element_ptr X, const size_t incX)`  

$$fzero : A \leftarrow 0.$$
- `template<class Field, class Randlter >`  
`void frand (const Field &F, Randlter &G, const size_t n, typename Field::Element_ptr X, const size_t incX)`  

$$frand : A \leftarrow random.$$
- `template<class Field >`  
`bool fiszero (const Field &F, const size_t n, typename Field::ConstElement_ptr X, const size_t incX)`  

$$fiszero : test\ X = 0.$$
- `template<class Field >`  
`bool fequal (const Field &F, const size_t n, typename Field::ConstElement_ptr X, const size_t incX, typename Field::ConstElement_ptr Y, const size_t incY)`  

$$fequal : test\ X = Y.$$
- `template<class Field >`  
`void fassign (const Field &F, const size_t N, typename Field::ConstElement_ptr Y, const size_t incY, typename Field::Element_ptr X, const size_t incX)`  

$$fassign : x \leftarrow y.$$
- `template<class Field >`  
`void fscaln (const Field &F, const size_t n, const typename Field::Element alpha, typename Field::Element_ptr X, const size_t incX)`  

$$fscaln\ x \leftarrow \alpha \cdot x.$$
- `template<class Field >`  
`void fscal (const Field &F, const size_t n, const typename Field::Element alpha, typename Field::ConstElement_ptr X, const size_t incX, typename Field::Element_ptr Y, const size_t incY)`

- $$fscal\ y \leftarrow \alpha \cdot x.$$
  - template<class [Field](#) >  
void [faxpy](#) (const [Field](#) &F, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, typename [Field::Element\\_ptr](#) Y, const size\_t incY)  
$$faxpy : y \leftarrow \alpha \cdot x + y.$$
- template<class [Field](#) >  
void [faxpby](#) (const [Field](#) &F, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) Y, const size\_t incY)  
$$faxpby : y \leftarrow \alpha \cdot x + \beta \cdot y.$$
- template<class [Field](#) >  
[Field::Element](#) [fdot](#) (const [Field](#) &F, const size\_t N, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, typename [Field::ConstElement\\_ptr](#) Y, const size\_t incY)  
$$fdot: \text{dot product } x^T y.$$
- template<class [Field](#) >  
[Field::Element](#) [fdot](#) (const [Field](#) &F, const size\_t N, typename [Field::ConstElement\\_ptr](#) x, const size\_t incx, typename [Field::ConstElement\\_ptr](#) y, const size\_t incy, const [ParSeqHelper::Sequential](#) seq)
- template<typename [Field](#) , class Cut , class Param >  
[Field::Element](#) [fdot](#) (const [Field](#) &F, const size\_t N, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, typename [Field::ConstElement\\_ptr](#) Y, const size\_t incY, const [ParSeqHelper::Parallel](#)< Cut, Param > par)
- template<class [Field](#) >  
void [fswap](#) (const [Field](#) &F, const size\_t N, typename [Field::Element\\_ptr](#) X, const size\_t incX, typename [Field::Element\\_ptr](#) Y, const size\_t incY)  
$$fswap: X \leftrightarrow Y.$$
- template<class [Field](#) >  
void [pfadd](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::Element\\_ptr](#) C, const size\_t ldc, const size\_t numths)
- template<class [Field](#) >  
void [pfsub](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::Element\\_ptr](#) C, const size\_t ldc, const size\_t numths)
- template<class [Field](#) >  
void [pfaddin](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::Element\\_ptr](#) C, const size\_t ldc, size\_t numths)
- template<class [Field](#) >  
void [pfsubin](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::Element\\_ptr](#) C, const size\_t ldc, size\_t numths)
- template<class [Field](#) >  
void [fadd](#) (const [Field](#) &F, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t inca, typename [Field::ConstElement\\_ptr](#) B, const size\_t incb, typename [Field::Element\\_ptr](#) C, const size\_t incc)
- template<class [Field](#) >  
void [fsub](#) (const [Field](#) &F, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t inca, typename [Field::ConstElement\\_ptr](#) B, const size\_t incb, typename [Field::Element\\_ptr](#) C, const size\_t incc)
- template<class [Field](#) >  
void [faddin](#) (const [Field](#) &F, const size\_t N, typename [Field::ConstElement\\_ptr](#) B, const size\_t incb, typename [Field::Element\\_ptr](#) C, const size\_t incc)
- template<class [Field](#) >  
void [fsubin](#) (const [Field](#) &F, const size\_t N, typename [Field::ConstElement\\_ptr](#) B, const size\_t incb, typename [Field::Element\\_ptr](#) C, const size\_t incc)
- template<class [Field](#) >  
void [fadd](#) (const [Field](#) &F, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t inca, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) B, const size\_t incb, typename [Field::Element\\_ptr](#) C, const size\_t incc)

### 13.112.1 Macro Definition Documentation

#### 13.112.1.1 \_\_FFLASFFPACK\_fflas\_fflas\_level1\_INL

```
#define __FFLASFFPACK_fflas_fflas_level1_INL
```

### 13.113 fflas\_level2.inl File Reference

```
#include "givaro/zring.h"
```

#### Namespaces

- namespace [FFLAS](#)

#### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_fflas\\_level2\\_INL](#)

#### Functions

- template<class [Field](#) >  
void [fassign](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
*fassign :  $A \leftarrow B$ .*
- template<class [Field](#) >  
void [fzero](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
*fzero :  $A \leftarrow 0$ .*
- template<class [Field](#) >  
void [fzero](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) shape, const [FFLAS\\_DIAG](#) diag, const size\_t n, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
*fzero :  $A \leftarrow 0$  for a triangular matrix.*
- template<class [Field](#) , class RandIter >  
void [frand](#) (const [Field](#) &F, RandIter &G, const size\_t m, const size\_t n, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
*frand :  $A \leftarrow \text{random}$ .*
- template<class [Field](#) >  
bool [fequal](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb)  
*fequal : test  $A = B$ .*
- template<class [Field](#) >  
bool [fiszero](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::ConstElement\\_ptr](#) A, const size\_t m, const size\_t n, const size\_t lda)  
*fiszero : test  $A = 0$ .*
- template<class [Field](#) >  
void [fidentity](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::Element\\_ptr](#) A, const size\_t lda, const typename [Field::Element](#) &d)  
*creates a diagonal matrix*
- template<class [Field](#) >  
void [fidentity](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
*creates a diagonal matrix*
- template<class [Field](#) >  
void [freduce](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
*freduce  $A \leftarrow A \bmod F$ .*
- template<class [Field](#) >  
void [freduce](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) uplo, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda)

*freduce for square symmetric matrices*

- template<class [Field](#) >  
void [freduce](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
*freduce*  $A \leftarrow B \bmod F$ .
- template<class [Field](#) , class [OtherElement\\_ptr](#) >  
void [finit](#) (const [Field](#) &F, const size\_t m, const size\_t n, const [OtherElement\\_ptr](#) B, const size\_t ldb, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
*finit*  $A \leftarrow B \bmod F$ .
- template<class [Field](#) , class [OtherElement\\_ptr](#) >  
void [finit](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
*finit* Initializes  $A$  in  $F^S$ .
- template<class [Field](#) , class [OtherElement\\_ptr](#) >  
void [fconvert](#) (const [Field](#) &F, const size\_t m, const size\_t n, [OtherElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb)  
*fconvert*  $A \leftarrow B \bmod F$ .
- template<class [Field](#) >  
void [fnegin](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
*fnegin*  $A \leftarrow -A$ .
- template<class [Field](#) >  
void [fneg](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
*fneg*  $A \leftarrow -B$ .
- template<class [Field](#) >  
void [fscaln](#) (const [Field](#) &F, const size\_t m, const size\_t n, const typename [Field::Element](#) alpha, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
*fscaln*  $A \leftarrow a \cdot A$ .
- template<class [Field](#) >  
void [fscal](#) (const [Field](#) &F, const size\_t m, const size\_t n, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) B, const size\_t ldb)  
*fscal*  $B \leftarrow a \cdot A$ .
- template<class [Field](#) >  
void [faxpy](#) (const [Field](#) &F, const size\_t m, const size\_t n, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) X, const size\_t ldx, typename [Field::Element\\_ptr](#) Y, const size\_t ldy)  
*faxpy* :  $y \leftarrow \alpha \cdot x + y$ .
- template<class [Field](#) >  
void [faxpby](#) (const [Field](#) &F, const size\_t m, const size\_t n, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) X, const size\_t ldx, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) Y, const size\_t ldy)  
*faxpby* :  $y \leftarrow \alpha \cdot x + \beta \cdot y$ .
- template<class [Field](#) >  
void [fmove](#) (const [Field](#) &F, const size\_t m, const size\_t n, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) B, const size\_t ldb)  
*fmove* :  $A \leftarrow B$  and  $B \leftarrow 0$ .
- template<class [Field](#) >  
void [fadd](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::Element\\_ptr](#) C, const size\_t ldc)  
*fadd* : matrix addition.
- template<class [Field](#) >  
void [fsub](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::Element\\_ptr](#) C, const size\_t ldc)  
*fsub* : matrix subtraction.
- template<class [Field](#) >  
void [fsubin](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::Element\\_ptr](#) C, const size\_t ldc)

- fsubin*  $C = C - B$
- template<class [Field](#) >  
void [fadd](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::Element\\_ptr](#) C, const size\_t ldc)  
*fadd* : matrix addition with scaling.
  - template<class [Field](#) >  
void [faddin](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::Element\\_ptr](#) C, const size\_t ldc)  
*faddin*
  - template<class [Field](#) >  
void [faddin](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) uplo, const size\_t N, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, typename [Field::Element\\_ptr](#) C, const size\_t ldc)  
*fadding* for symmetric matrices
  - template<class [Field](#) >  
[Field::Element\\_ptr](#) [fgemv](#) (const [Field](#) &F, const [FFLAS\\_TRANSPOSE](#) TransA, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) Y, const size\_t incY)  
*finite prime Field GEneral Matrix Vector multiplication.*
  - template<class [Field](#) >  
void [fger](#) (const [Field](#) &F, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) x, const size\_t incx, typename [Field::ConstElement\\_ptr](#) y, const size\_t incy, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
*fger*: rank one update of a general matrix
  - template<class [Field](#) >  
void [ftrsv](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) Uplo, const [FFLAS\\_TRANSPOSE](#) TransA, const [FFLAS\\_DIAG](#) Diag, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) X, int incX)  
*ftrsv*: *TR*angular System solve with Vector Computes  $X \leftarrow \text{op}(A^{-1})X$
  - template<class [Field](#) >  
size\_t [bitsize](#) (const [Field](#) &F, size\_t M, size\_t N, const typename [Field::ConstElement\\_ptr](#) A, size\_t lda)  
*bitsize*: Computes the largest bitsize of the matrix' coefficients.
  - template<> size\_t [bitsize](#)< [Givaro::ZRing](#)< [Givaro::Integer](#) > > (const [Givaro::ZRing](#)< [Givaro::Integer](#) > &F, size\_t M, size\_t N, const [Givaro::Integer](#) \*A, size\_t lda)
  - template<class [Field](#) >  
void [ftmrv](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) Uplo, const [FFLAS\\_TRANSPOSE](#) TransA, const [FFLAS\\_DIAG](#) Diag, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) X, int incX)  
*ftsm*: *TR*angular Matrix Vector prodcut Computes  $X \leftarrow \text{op}(A)X$

### 13.113.1 Macro Definition Documentation

#### 13.113.1.1 \_\_FFLASFFPACK\_fflas\_fflas\_level2\_INL

```
#define __FFLASFFPACK_fflas_fflas_level2_INL
```

### 13.114 fflas\_level3.inl File Reference

```
#include "fflas_bounds.inl"
#include "fflas_helpers.inl"
#include "fflas-ffpack/paladin/parallel.h"
```



## Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::Protected](#)

## Macros

- `#define` [\\_\\_FFLASFFPACK\\_fflas\\_fflas\\_level3\\_INL](#)
- `#define` [\\_\\_FFLAS\\_\\_TRSM\\_READONLY](#)

## Functions

- `template<class Field >`  
void [MatF2MatD\\_Triangular](#) (const [Field](#) &F, Givaro::DoubleDomain::Element\_ptr S, const size\_t lds, typename [Field::ConstElement\\_ptr](#) const E, const size\_t lde, const size\_t m, const size\_t n)
- `template<class Field >`  
void [MatF2MatFI\\_Triangular](#) (const [Field](#) &F, Givaro::FloatDomain::Element\_ptr S, const size\_t lds, typename [Field::ConstElement\\_ptr](#) const E, const size\_t lde, const size\_t m, const size\_t n)
- `template<class Field >`  
void [ftrsm](#) (const [Field](#) &F, const [FFLAS\\_SIDE](#) Side, const [FFLAS\\_UPLO](#) Uplo, const [FFLAS\\_TRANSPOSE](#) TransA, const [FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) B, const size\_t ldb)  
*ftrsm: **TRI**angular **S**ystem solve with **M**atrix.*
- `template<class Field >`  
void [ftrmm](#) (const [Field](#) &F, const [FFLAS\\_SIDE](#) Side, const [FFLAS\\_UPLO](#) Uplo, const [FFLAS\\_TRANSPOSE](#) TransA, const [FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) B, const size\_t ldb)  
*ftrmm: **TRI**angular **M**atrix **M**ultiply.*
- `template<class Field >`  
void [ftrmm](#) (const [Field](#) &F, const [FFLAS\\_SIDE](#) Side, const [FFLAS\\_UPLO](#) Uplo, const [FFLAS\\_TRANSPOSE](#) TransA, const [FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc)  
*ftrmm: **TRI**angular **M**atrix **M**ultiply with 3 operands Computes  $C \leftarrow \alpha op(A)B + beta C$  or  $C \leftarrow \alpha B op(A) + beta C$ .*
- `template<class Field >`  
[Field::Element\\_ptr](#) [fsyrk](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) UpLo, const [FFLAS\\_TRANSPOSE](#) trans, const size\_t n, const size\_t k, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc)  
*fsyrk: Symmetric Rank K update*
- `template<class Field, typename FieldTrait >`  
[Field::Element\\_ptr](#) [fsyrk\\_strassen](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) UpLo, const [FFLAS\\_TRANSPOSE](#) trans, const size\_t N, const size\_t K, const typename [Field::Element](#) y1, const typename [Field::Element](#) y2, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc, [MMHelper](#)< [Field](#), [MMHelperAlgo::Winograd](#), FieldTrait > &H)
- `template<class Field >`  
[Field::Element\\_ptr](#) [fsyr2k](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) UpLo, const [FFLAS\\_TRANSPOSE](#) trans, const size\_t n, const size\_t k, const typename [Field::Element](#) alpha, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc)  
*fsyr2k: Symmetric Rank 2K update*
- `template<class Field >`  
[Field::Element\\_ptr](#) [fsyrk](#) (const [Field](#) &F, const [FFLAS\\_UPLO](#) UpLo, const [FFLAS\\_TRANSPOSE](#) trans, const size\_t n, const size\_t k, const typename [Field::Element](#) alpha, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) D, const size\_t incD, const typename [Field::Element](#) beta, typename [Field::Element\\_ptr](#) C, const size\_t ldc, const size\_t threshold=[\\_\\_FFLASFFPACK\\_FSYRK\\_THRESHOLD](#))  
*fsyrk: Symmetric Rank K update with diagonal scaling*

- `template<class Field >`  
`Field::Element_ptr fsyrk` (const `Field` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` N, const `size_t` K, const typename `Field::Element` alpha, typename `Field::Element_ptr` A, const `size_t` lda, typename `Field::ConstElement_ptr` D, const `size_t` incD, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc, const `ParSeqHelper::Sequential` seq, const `size_t` threshold)
- `template<class Field , class Cut , class Param >`  
`Field::Element_ptr fsyrk` (const `Field` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` N, const `size_t` K, const typename `Field::Element` alpha, typename `Field::Element_ptr` A, const `size_t` lda, typename `Field::ConstElement_ptr` D, const `size_t` incD, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc, const `ParSeqHelper::Parallel`< Cut, Param > par, const `size_t` threshold)
- `template<class Field >`  
`Field::Element_ptr fsyrk` (const `Field` &F, const `FFLAS_UPLO` UpLo, const `FFLAS_TRANSPOSE` trans, const `size_t` n, const `size_t` k, const typename `Field::Element` alpha, typename `Field::Element_ptr` A, const `size_t` lda, typename `Field::ConstElement_ptr` D, const `size_t` incD, const `std::vector< bool >` &two←Block, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc, const `size_t` threshold=\_\_FFLASFFPACK\_FSYRK\_THRESHOLD)  
*fsyrk: Symmetric Rank K update with diagonal scaling*
- `template<typename Field >`  
`Field::Element_ptr fgemm` (const `Field` &F, const `FFLAS_TRANSPOSE` ta, const `FFLAS_TRANSPOSE` tb, const `size_t` m, const `size_t` n, const `size_t` k, const typename `Field::Element` alpha, typename `Field::ConstElement_ptr` A, const `size_t` lda, typename `Field::ConstElement_ptr` B, const `size_t` ldb, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc)  
*fgemm: Field GENeral Matrix Multiply.*
- `template<typename Field >`  
`Field::Element_ptr fgemm` (const `Field` &F, const `FFLAS_TRANSPOSE` ta, const `FFLAS_TRANSPOSE` tb, const `size_t` m, const `size_t` n, const `size_t` k, const typename `Field::Element` alpha, typename `Field::ConstElement_ptr` A, const `size_t` lda, typename `Field::ConstElement_ptr` B, const `size_t` ldb, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc, const `ParSeqHelper::Sequential` seq)
- `template<typename Field , class Cut , class Param >`  
`Field::Element_ptr fgemm` (const `Field` &F, const `FFLAS_TRANSPOSE` ta, const `FFLAS_TRANSPOSE` tb, const `size_t` m, const `size_t` n, const `size_t` k, const typename `Field::Element` alpha, typename `Field::ConstElement_ptr` A, const `size_t` lda, typename `Field::ConstElement_ptr` B, const `size_t` ldb, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc, const `ParSeqHelper::Parallel`< Cut, Param > par)
- `template<typename Field >`  
`Field::Element_ptr pfgemm` (const `Field` &F, const `FFLAS_TRANSPOSE` ta, const `FFLAS_TRANSPOSE` tb, const `size_t` m, const `size_t` n, const `size_t` k, const typename `Field::Element` alpha, typename `Field::ConstElement_ptr` A, const `size_t` lda, typename `Field::ConstElement_ptr` B, const `size_t` ldb, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc, `size_t` numthreads=0)
- `template<class Field >`  
`Field::Element * pfgemm_1D_rec` (const `Field` &F, const `FFLAS_TRANSPOSE` ta, const `FFLAS_TRANSPOSE` tb, const `size_t` m, const `size_t` n, const `size_t` k, const typename `Field::Element` alpha, const typename `Field::Element_ptr` A, const `size_t` lda, const typename `Field::Element_ptr` B, const `size_t` ldb, const typename `Field::Element` beta, typename `Field::Element` \*C, const `size_t` ldc, `size_t` seuil)
- `template<class Field >`  
`Field::Element * pfgemm_2D_rec` (const `Field` &F, const `FFLAS_TRANSPOSE` ta, const `FFLAS_TRANSPOSE` tb, const `size_t` m, const `size_t` n, const `size_t` k, const typename `Field::Element` alpha, const typename `Field::Element_ptr` A, const `size_t` lda, const typename `Field::Element_ptr` B, const `size_t` ldb, const typename `Field::Element` beta, typename `Field::Element` \*C, const `size_t` ldc, `size_t` seuil)
- `template<class Field >`  
`Field::Element * pfgemm_3D_rec` (const `Field` &F, const `FFLAS_TRANSPOSE` ta, const `FFLAS_TRANSPOSE` tb, const `size_t` m, const `size_t` n, const `size_t` k, const typename `Field::Element` alpha, const typename `Field::Element_ptr` A, const `size_t` lda, const typename `Field::Element_ptr` B, const `size_t` ldb, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc, `size_t` seuil, `size_t` \*x)

- `template<class Field >`  
`Field::Element_ptr pfgemm_3D_rec2` (const `Field` &F, const `FFLAS_TRANSPOSE` ta, const `FFLAS_TRANSPOSE` tb, const `size_t` m, const `size_t` n, const `size_t` k, const typename `Field::Element` alpha, const typename `Field::Element_ptr` A, const `size_t` lda, const typename `Field::Element_ptr` B, const `size_t` ldb, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc, `size_t` seuil, `size_t` \*x)
- `template<class Field >`  
`Field::Element_ptr fsquare` (const `Field` &F, const `FFLAS_TRANSPOSE` ta, const `size_t` n, const typename `Field::Element` alpha, typename `Field::ConstElement_ptr` A, const `size_t` lda, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc)

*fsquare: Squares a matrix.*

### 13.114.1 Macro Definition Documentation

#### 13.114.1.1 \_\_FFLASFFPACK\_fflas\_fflas\_level3\_INL

```
#define __FFLASFFPACK_fflas_fflas_level3_INL
```

#### 13.114.1.2 \_\_FFLAS\_\_TRSM\_READONLY

```
#define __FFLAS__TRSM_READONLY
```

## 13.115 fflas\_pfgemm.inl File Reference

```
#include "fflas-ffpack/paladin/blockcuts.inl"
#include "fflas-ffpack/paladin/parallel.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/paladin/pfgemm_variants.inl"
```

### Namespaces

- namespace `FFLAS`

### Macros

- `#define __FFLASFFPACK_fflas_pfgemm_INL`
- `#define __FFLASFFPACK_SEQPARTHRESHOLD 220`
- `#define __FFLASFFPACK_DIMKPENALTY 1`

### Functions

- `template<class Field , class ModeTrait , class Strat , class Param >`  
`std::enable_if<!std::is_same< ModeTrait, ModeCategories::ConvertTo< ElementCategories::RNSElementTag > >::value, typename Field::Element_ptr >::type` `fgemm` (const `Field` &F, const `FFLAS::FFLAS_TRANSPOSE` ta, const `FFLAS::FFLAS_TRANSPOSE` tb, const `size_t` m, const `size_t` n, const `size_t` k, const typename `Field::Element` alpha, typename `Field::ConstElement_ptr` A, const `size_t` lda, typename `Field::ConstElement_ptr` B, const `size_t` ldb, const typename `Field::Element` beta, typename `Field::Element_ptr` C, const `size_t` ldc, `MMHelper`< `Field`, `MMHelperAlgo::Winograd`, `ModeTrait`, `ParSeqHelper::Parallel`< `Strat`, `Param` > > &H)

### 13.115.1 Macro Definition Documentation

#### 13.115.1.1 \_\_FFLASFFPACK\_fflas\_pfgemm\_INL

```
#define __FFLASFFPACK_fflas_pfgemm_INL
```

#### 13.115.1.2 \_\_FFLASFFPACK\_SEQPARTHRESHOLD

```
#define __FFLASFFPACK_SEQPARTHRESHOLD 220
```

### 13.115.1.3 \_\_FFLASFFPACK\_DIMKPENALTY

```
#define __FFLASFFPACK_DIMKPENALTY 1
```

## 13.116 fflas\_pftrsm.inl File Reference

```
#include "fflas-ffpack/paladin/parallel.h"
```

### Namespaces

- namespace [FFLAS](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_pftrsm\\_INL](#)
- #define [PTRSM\\_HYBRID\\_THRESHOLD](#) 256

### Functions

- template<class [Field](#), class Cut, class Param >  
[Field::Element\\_ptr](#) ftrsm (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const [FFLAS::FFLAS\\_UPLO](#) UpLo, const [FFLAS::FFLAS\\_TRANSPOSE](#) TA, const [FFLAS::FFLAS\\_DIAG](#) Diag, const size\_t m, const size\_t n, const typename [Field::Element](#) alpha, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) B, const size\_t ldb, [TRSMHelper](#)< [StructureHelper::Iterative](#), [ParSeqHelper::Parallel](#)< Cut, Param > > &H)
- template<class [Field](#), class Cut, class Param >  
[Field::Element\\_ptr](#) ftrsm (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const [FFLAS::FFLAS\\_UPLO](#) UpLo, const [FFLAS::FFLAS\\_TRANSPOSE](#) TA, const [FFLAS::FFLAS\\_DIAG](#) Diag, const size\_t m, const size\_t n, const typename [Field::Element](#) alpha, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) B, const size\_t ldb, [TRSMHelper](#)< [StructureHelper::Hybrid](#), [ParSeqHelper::Parallel](#)< Cut, Param > > &H)

## 13.116.1 Macro Definition Documentation

### 13.116.1.1 \_\_FFLASFFPACK\_fflas\_pftrsm\_INL

```
#define __FFLASFFPACK_fflas_pftrsm_INL
```

### 13.116.1.2 PTRSM\_HYBRID\_THRESHOLD

```
#define PTRSM_HYBRID_THRESHOLD 256
```

## 13.117 fflas\_simd.h File Reference

```
#include "fflas-ffpack/utils/fflas_intrinsic.h"
#include <iostream>
#include <type_traits>
#include <limits>
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/utils/debug.h"
#include "fflas-ffpack/utils/align-allocator.h"
#include <vector>
#include "givaro/givtypestring.h"
#include <fflas-ffpack/fflas/fflas_simd/simd_modular.inl>
```

## Data Structures

- struct [support\\_simd](#)< T >
- struct [is\\_simd](#)< T >
- struct [NoSimd](#)< T >
- struct [SimdChooser](#)< T, bool, bool >
- struct [SimdChooser](#)< T, false, b >
- struct [SimdChooser](#)< T, true, false >
- struct [SimdChooser](#)< T, true, true >

## Namespaces

- namespace [FFLAS](#)

## Macros

- `#define` [SIMD\\_INT](#) 1
- `#define` [INLINE](#) inline
- `#define` [CONST](#)
- `#define` [PURE](#)
- `#define` [NORML\\_MOD](#)(C, P, NEGP, MIN, MAX, Q, T)
- `#define` [FLOAT\\_MOD](#)(C, P, INVP, Q)

## Typedefs

- `template<class T >`  
using [Simd](#) = typename [SimdChooser](#)< T >::value

## 13.117.1 Macro Definition Documentation

### 13.117.1.1 SIMD\_INT

```
#define SIMD_INT 1
```

### 13.117.1.2 INLINE

```
#define INLINE inline
```

### 13.117.1.3 CONST

```
#define CONST
```

### 13.117.1.4 PURE

```
#define PURE
```

### 13.117.1.5 NORML\_MOD

```
#define NORML_MOD(
 C,
 P,
 NEGP,
 MIN,
 MAX,
 Q,
 T)
```

#### Value:

```
{
 \
 Q = greater(C, MAX);
 \
}
```

```

 T = lesser(C, MIN);
 \
 Q = vand(Q, NEGP);
 \
 T = vand(T, P);
 \
 Q = vor(Q, T);
 \
 C = add(C, Q);
 \
}

```

### 13.117.1.6 FLOAT\_MOD

```

#define FLOAT_MOD(
 C,
 P,
 INVP,
 Q)

```

**Value:**

```

{
 \
 Q = mul(C, INVP);
 \
 Q = floor(Q);
 \
 C = fnmadd(C, Q, P);
 \
}

```

## 13.117.2 Typedef Documentation

### 13.117.2.1 Simd

```

template<class T >
using Simd = typename SimdChooser<T>::value

```

## 13.118 simd.doxy File Reference

## 13.119 simd128.inl File Reference

```

#include "simd128_float.inl"
#include "simd128_double.inl"

```

### Data Structures

- struct [Simd128i\\_base](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_ffpack\\_utils\\_simd128\\_INL](#)

### Typedefs

- template<class T >  
using [Simd128](#) = [Simd128\\_impl](#)< std::is\_arithmetic< T >::value, std::is\_integral< T >::value, std::is\_↵  
signed< T >::value, sizeof(T)>

## 13.119.1 Macro Definition Documentation

### 13.119.1.1 \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_INL

```

#define __FFLASFFPACK_fflas_ffpack_utils_simd128_INL

```

## 13.119.2 Typedef Documentation

### 13.119.2.1 Simd128

```
template<class T >
using Simd128 = Simd128_impl<std::is_arithmetic<T>::value, std::is_integral<T>::value, std::is_signed<T>::value, sizeof(T)>
```

## 13.120 simd128\_double.inl File Reference

```
#include "givaro/givtypestring.h"
#include "fflas-ffpack/utils/align-allocator.h"
#include <vector>
#include <type_traits>
```

### Data Structures

- struct [Simd128\\_impl< true, false, true, 8 >](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_ffpack\\_utils\\_simd128\\_double\\_INL](#)

### 13.120.1 Macro Definition Documentation

#### 13.120.1.1 \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_double\_INL

```
#define __FFLASFFPACK_fflas_ffpack_utils_simd128_double_INL
```

## 13.121 simd128\_float.inl File Reference

```
#include "givaro/givtypestring.h"
#include "fflas-ffpack/utils/align-allocator.h"
#include <vector>
#include <type_traits>
```

### Data Structures

- struct [Simd128\\_impl< true, false, true, 4 >](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_ffpack\\_utils\\_simd128\\_float\\_INL](#)

### 13.121.1 Macro Definition Documentation

#### 13.121.1.1 \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_float\_INL

```
#define __FFLASFFPACK_fflas_ffpack_utils_simd128_float_INL
```

## 13.122 simd128\_int16.inl File Reference

```
#include "givaro/givtypestring.h"
#include "fflas-ffpack/utils/align-allocator.h"
#include <vector>
#include <type_traits>
```

**Data Structures**

- struct [Simd128\\_impl< true, true, true, 2 >](#)
- union [Simd128\\_impl< true, true, true, 2 >::Converter](#)
- struct [Simd128\\_impl< true, true, false, 2 >](#)
- union [Simd128\\_impl< true, true, false, 2 >::Converter](#)

**Macros**

- `#define __FFLASFFPACK_fflas_ffpack_utils_simd128_int16_INL`

**13.122.1 Macro Definition Documentation****13.122.1.1 \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_int16\_INL**

```
#define __FFLASFFPACK_fflas_ffpack_utils_simd128_int16_INL
```

**13.123 simd128\_int32.inl File Reference**

```
#include "givaro/givtypestring.h"
#include "fflas-ffpack/utils/align-allocator.h"
#include <vector>
#include <type_traits>
```

**Data Structures**

- struct [Simd128\\_impl< true, true, true, 4 >](#)
- union [Simd128\\_impl< true, true, true, 4 >::Converter](#)
- struct [Simd128\\_impl< true, true, false, 4 >](#)
- union [Simd128\\_impl< true, true, false, 4 >::Converter](#)

**Macros**

- `#define __FFLASFFPACK_fflas_ffpack_utils_simd128_int32_INL`

**13.123.1 Macro Definition Documentation****13.123.1.1 \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_int32\_INL**

```
#define __FFLASFFPACK_fflas_ffpack_utils_simd128_int32_INL
```

**13.124 simd128\_int64.inl File Reference**

```
#include "givaro/givtypestring.h"
#include "fflas-ffpack/utils/align-allocator.h"
#include "fflas-ffpack/utils/bit_manipulation.h"
#include <vector>
#include <type_traits>
```

**Data Structures**

- struct [Simd128\\_impl< true, true, true, 8 >](#)
- union [Simd128\\_impl< true, true, true, 8 >::Converter](#)
- struct [Simd128\\_impl< true, true, false, 8 >](#)
- union [Simd128\\_impl< true, true, false, 8 >::Converter](#)



**Macros**

- `#define \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_int64\_INL`
- `#define vect\_t Simd128\_impl<true,true,true,8>::vect_t`

**13.124.1 Macro Definition Documentation****13.124.1.1 [\\_\\_FFLASFFPACK\\_fflas\\_ffpack\\_utils\\_simd128\\_int64\\_INL](#)**

```
#define __FFLASFFPACK_fflas_ffpack_utils_simd128_int64_INL
```

**13.124.1.2 [vect\\_t](#)**

```
#define vect_t Simd128_impl<true,true,true,8>::vect_t
```

**13.125 simd256.inl File Reference**

```
#include "simd256_float.inl"
#include "simd256_double.inl"
```

**Data Structures**

- struct [Simd256fp\\_base](#)
- struct [Simd256i\\_base](#)

**Macros**

- `#define \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_INL`

**Typedefs**

- `template<class T >`  
`using Simd256 = Simd256\_impl< std::is_arithmetic< T >::value, std::is_integral< T >::value, std::is_↵`  
`signed< T >::value, sizeof(T)>`

**13.125.1 Macro Definition Documentation****13.125.1.1 [\\_\\_FFLASFFPACK\\_fflas\\_ffpack\\_utils\\_simd256\\_INL](#)**

```
#define __FFLASFFPACK_fflas_ffpack_utils_simd256_INL
```

**13.125.2 Typedef Documentation****13.125.2.1 [Simd256](#)**

```
template<class T >
using Simd256 = Simd256_impl<std::is_arithmetic<T>::value, std::is_integral<T>::value, std↵
::is_signed<T>::value, sizeof(T)>
```

**13.126 simd256\_double.inl File Reference**

```
#include "givaro/givtypestring.h"
#include "fflas-ffpack/utils/align-allocator.h"
#include <vector>
#include <type_traits>
```

**Data Structures**

- struct [Simd256\\_impl< true, false, true, 8 >](#)
- union [Simd256\\_impl< true, false, true, 8 >::Converter](#)

**Macros**

- [#define \\_\\_FFLASFFPACK\\_fflas\\_ffpack\\_utils\\_simd256\\_double\\_INL](#)

**13.126.1 Macro Definition Documentation****13.126.1.1 \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_double\_INL**

```
#define __FFLASFFPACK_fflas_ffpack_utils_simd256_double_INL
```

**13.127 simd256\_float.inl File Reference**

```
#include "givaro/givtypestring.h"
#include "fflas-ffpack/utils/align-allocator.h"
#include <vector>
#include <type_traits>
```

**Data Structures**

- struct [Simd256\\_impl< true, false, true, 4 >](#)

**Macros**

- [#define \\_\\_FFLASFFPACK\\_fflas\\_ffpack\\_utils\\_simd256\\_float\\_INL](#)

**13.127.1 Macro Definition Documentation****13.127.1.1 \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_float\_INL**

```
#define __FFLASFFPACK_fflas_ffpack_utils_simd256_float_INL
```

**13.128 simd256\_int16.inl File Reference**

```
#include "givaro/givtypestring.h"
#include "fflas-ffpack/utils/align-allocator.h"
#include <vector>
#include <type_traits>
```

**Data Structures**

- struct [Simd256\\_impl< true, true, true, 2 >](#)
- union [Simd256\\_impl< true, true, true, 2 >::Converter](#)
- struct [Simd256\\_impl< true, true, false, 2 >](#)
- union [Simd256\\_impl< true, true, false, 2 >::Converter](#)

**Macros**

- [#define \\_\\_FFLASFFPACK\\_fflas\\_ffpack\\_utils\\_simd256\\_int16\\_INL](#)

## 13.128.1 Macro Definition Documentation

### 13.128.1.1 \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_int16\_INL

```
#define __FFLASFFPACK_fflas_ffpack_utils_simd256_int16_INL
```

## 13.129 simd256\_int32.inl File Reference

```
#include "givaro/givtypestring.h"
#include "fflas-ffpack/utils/align-allocator.h"
#include <vector>
#include <type_traits>
```

### Data Structures

- struct [Simd256\\_impl< true, true, true, 4 >](#)
- union [Simd256\\_impl< true, true, true, 4 >::Converter](#)
- struct [Simd256\\_impl< true, true, false, 4 >](#)
- union [Simd256\\_impl< true, true, false, 4 >::Converter](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_ffpack\\_utils\\_simd256\\_int32\\_INL](#)

## 13.129.1 Macro Definition Documentation

### 13.129.1.1 \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_int32\_INL

```
#define __FFLASFFPACK_fflas_ffpack_utils_simd256_int32_INL
```

## 13.130 simd256\_int64.inl File Reference

```
#include "givaro/givtypestring.h"
#include "fflas-ffpack/utils/align-allocator.h"
#include "fflas-ffpack/utils/bit_manipulation.h"
#include <vector>
#include <type_traits>
```

### Data Structures

- struct [Simd256\\_impl< true, true, true, 8 >](#)
- union [Simd256\\_impl< true, true, true, 8 >::Converter](#)
- struct [Simd256\\_impl< true, true, false, 8 >](#)
- union [Simd256\\_impl< true, true, false, 8 >::Converter](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_ffpack\\_utils\\_simd256\\_int64\\_INL](#)
- #define [vect\\_t Simd256\\_impl<true, true, true, 8>::vect\\_t](#)

## 13.130.1 Macro Definition Documentation

### 13.130.1.1 \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_int64\_INL

```
#define __FFLASFFPACK_fflas_ffpack_utils_simd256_int64_INL
```

### 13.130.1.2 vect\_t

```
#define vect_t Simd256_impl<true, true, true, 8>::vect_t
```

## 13.131 simd512.inl File Reference

```
#include "simd512_float.inl"
#include "simd512_double.inl"
#include "simd512_int64.inl"
```

### Data Structures

- struct [Simd512i\\_base](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_simd512\\_INL](#)

### Typedefs

- template<class T >  
using [Simd512](#) = [Simd512\\_impl](#)< std::is\_arithmetic< T >::value, std::is\_integral< T >::value, std::is\_↵  
signed< T >::value, sizeof(T)>

### 13.131.1 Macro Definition Documentation

#### 13.131.1.1 \_\_FFLASFFPACK\_simd512\_INL

```
#define __FFLASFFPACK_simd512_INL
```

### 13.131.2 Typedef Documentation

#### 13.131.2.1 Simd512

```
template<class T >
using Simd512 = Simd512_impl<std::is_arithmetic<T>::value, std::is_integral<T>::value, std↵
::is_signed<T>::value, sizeof(T)>
```

## 13.132 simd512\_double.inl File Reference

```
#include "givaro/givtypestring.h"
#include "fflas-ffpack/utils/align-allocator.h"
#include <vector>
#include <type_traits>
```

### Data Structures

- struct [Simd512\\_impl](#)< true, false, true, 8 >

### Macros

- #define [\\_\\_FFLASFFPACK\\_simd512\\_double\\_INL](#)

### 13.132.1 Macro Definition Documentation

#### 13.132.1.1 \_\_FFLASFFPACK\_simd512\_double\_INL

```
#define __FFLASFFPACK_simd512_double_INL
```

## 13.133 simd512\_float.inl File Reference

```
#include "givaro/givtypestring.h"
#include "fflas-ffpack/utils/align-allocator.h"
#include <vector>
#include <type_traits>
```

### Data Structures

- struct [Simd512\\_impl< true, false, true, 4 >](#)

### Macros

- [#define \\_\\_FFLASFFPACK\\_simd512\\_float\\_INL](#)

### 13.133.1 Macro Definition Documentation

#### 13.133.1.1 \_\_FFLASFFPACK\_simd512\_float\_INL

```
#define __FFLASFFPACK_simd512_float_INL
```

## 13.134 simd512\_int32.inl File Reference

```
#include "fflas-ffpack/fflas/fflas_simd/simd512_int64.inl"
#include "givaro/givtypestring.h"
#include "fflas-ffpack/utils/align-allocator.h"
#include <vector>
#include <type_traits>
```

### Data Structures

- struct [Simd256\\_impl< true, true, true, 4 >](#)
- union [Simd256\\_impl< true, true, true, 4 >::Converter](#)
- struct [Simd256\\_impl< true, true, false, 4 >](#)
- union [Simd256\\_impl< true, true, false, 4 >::Converter](#)

### Macros

- [#define \\_\\_FFLASFFPACK\\_simd512\\_int32\\_INL](#)

### 13.134.1 Macro Definition Documentation

#### 13.134.1.1 \_\_FFLASFFPACK\_simd512\_int32\_INL

```
#define __FFLASFFPACK_simd512_int32_INL
```

## 13.135 simd512\_int64.inl File Reference

```
#include "givaro/givtypestring.h"
#include "fflas-ffpack/utils/align-allocator.h"
#include "fflas-ffpack/utils/bit_manipulation.h"
#include <vector>
#include <type_traits>
```

## Data Structures

- struct [Simd512\\_impl](#)< true, true, true, 8 >
- union [Simd512\\_impl](#)< true, true, true, 8 >::Converter
- struct [Simd512\\_impl](#)< true, true, false, 8 >
- union [Simd512\\_impl](#)< true, true, false, 8 >::Converter

## Macros

- `#define` [\\_simd512\\_int64\\_INL](#)
- `#define` [vect\\_t](#) [Simd512\\_impl](#)<true, true, true, 8>::vect\_t

## 13.135.1 Macro Definition Documentation

### 13.135.1.1 [\\_simd512\\_int64\\_INL](#)

```
#define _simd512_int64_INL
```

### 13.135.1.2 [vect\\_t](#)

```
#define vect_t Simd512_impl<true, true, true, 8>::vect_t
```

## 13.136 [simd\\_modular.inl](#) File Reference

### Data Structures

- class [FieldSimd](#)< [\\_Field](#) >

## 13.137 [fflas\\_sparse.h](#) File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/config.h"
#include "fflas-ffpack/config-blas.h"
#include "fflas-ffpack/paladin/parallel.h"
#include <recint/recint.h>
#include <givaro/udl.h>
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/field/field-traits.h"
#include "fflas-ffpack/fflas/fflas_bounds.inl"
#include "fflas-ffpack/utils/fflas_memory.h"
#include <type_traits>
#include <vector>
#include <iostream>
#include "fflas-ffpack/fflas/fflas_sparse/sparse_matrix_traits.h"
#include "fflas-ffpack/fflas/fflas_sparse/utils.h"
#include "fflas-ffpack/fflas/fflas_sparse/csr.h"
#include "fflas-ffpack/fflas/fflas_sparse/coo.h"
#include "fflas-ffpack/fflas/fflas_sparse/ell.h"
#include "fflas-ffpack/fflas/fflas_sparse/sell.h"
#include "fflas-ffpack/fflas/fflas_sparse/csr_hyb.h"
#include "fflas-ffpack/fflas/fflas_sparse/ell_simd.h"
#include "fflas-ffpack/fflas/fflas_sparse/hyb_zo.h"
#include "fflas-ffpack/fflas/fflas_sparse.inl"
#include "fflas-ffpack/fflas/fflas_sparse/read_sparse.h"
```

## Data Structures

- struct [HelperFlag](#)
- struct [CsrMat< Field >](#)
- struct [CooMat< Field >](#)
- struct [EllMat< Field >](#)
- struct [SpMat< Field, flag >](#)

## Namespaces

- namespace [MKL\\_CONFIG](#)
- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details](#)

## Macros

- `#define` [index\\_t](#) `uint32_t`
- `#define` [ROUND\\_DOWN](#)(x, s) `((x) & ~((s)-1))`
- `#define` [\\_\\_FFLASFFPACK\\_CACHE\\_LINE\\_SIZE](#) `64`
- `#define` [assume\\_aligned](#)(pout, pin, v) `decltype(pin) pout = pin;`
- `#define` [DENSE\\_THRESHOLD](#) `0.5`

## Enumerations

- enum class [SparseMatrix\\_t](#) {  
[CSR](#) , [CSR\\_ZO](#) , [CSC](#) , [CSC\\_ZO](#) ,  
[COO](#) , [COO\\_ZO](#) , [ELL](#) , [ELL\\_ZO](#) ,  
[SELL](#) , [SELL\\_ZO](#) , [ELL\\_simd](#) , [ELL\\_simd\\_ZO](#) ,  
[CSR\\_HYB](#) , [HYB\\_ZO](#) }

## Functions

- template<class [Field](#) >  
void [init\\_y](#) (const [Field](#) &F, const `size_t` m, const typename [Field::Element](#) b, typename [Field::Element\\_ptr](#) y)
- template<class [Field](#) >  
void [init\\_y](#) (const [Field](#) &F, const `size_t` m, const `size_t` n, const typename [Field::Element](#) b, typename [Field::Element\\_ptr](#) y, const `int` ldy)
- template<class [Field](#) , class [SM](#) , class [FC](#) , class [MZO](#) >  
std::enable\_if<!(std::is\_same< typename [ElementTraits](#)< typename [Field::Element](#) >::value, [ElementCategories::MachineFloat](#) >::value)||std::is\_same< typename [ElementTraits](#)< typename [Field::Element](#) >::value, [ElementCategories::MachineIntTag](#) >::value)>::type [fspmv\\_dispatch](#) (const [Field](#) &F, const [SM](#) &A, typename [Field::ConstElement\\_ptr](#) x, typename [Field::Element\\_ptr](#) y, [FC](#) fc, [MZO](#) mzo)
- template<class [Field](#) , class [SM](#) , class [FC](#) , class [MZO](#) >  
std::enable\_if< std::is\_same< typename [ElementTraits](#)< typename [Field::Element](#) >::value, [ElementCategories::MachineFloat](#) >::value)||std::is\_same< typename [ElementTraits](#)< typename [Field::Element](#) >::value, [ElementCategories::MachineIntTag](#) >::value >::type [fspmv\\_dispatch](#) (const [Field](#) &F, const [SM](#) &A, typename [Field::ConstElement\\_ptr](#) x, typename [Field::Element\\_ptr](#) y, [FC](#) fc, [MZO](#) mzo)
- template<class [Field](#) , class [SM](#) >  
void [fspmv](#) (const [Field](#) &F, const [SM](#) &A, typename [Field::ConstElement\\_ptr](#) x, typename [Field::Element\\_ptr](#) y, [FieldCategories::GenericTag](#), [NotZOSparseMatrix](#))
- template<class [Field](#) , class [SM](#) >  
std::enable\_if< [isSparseMatrixSimdFormat](#)< [Field](#), [SM](#) >::value >::type [fspmv](#) (const [Field](#) &F, const [SM](#) &A, typename [Field::ConstElement\\_ptr](#) x, typename [Field::Element\\_ptr](#) y, [FieldCategories::UnparametricTag](#), [NotZOSparseMatrix](#))
- template<class [Field](#) , class [SM](#) >  
std::enable\_if< [isSparseMatrixSimdFormat](#)< [Field](#), [SM](#) >::value >::type [fspmv](#) (const [Field](#) &F, const [SM](#) &A, typename [Field::ConstElement\\_ptr](#) x, typename [Field::Element\\_ptr](#) y, [FieldCategories::UnparametricTag](#), [NotZOSparseMatrix](#))

- `template<class Field , class SM >`  
`std::enable_if< !isSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::ModularTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< isSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::ModularTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`void fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::GenericTag, ZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< !isSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag, ZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< isSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag, ZOSparseMatrix)`
- `template<class Field , class SM >`  
`void fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::ModularTag, std::true_type)`
- `template<class Field , class SM , class FCat , class MZO >`  
`std::enable_if< !(std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineFloat >::value) | std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineIntTag >::value) >::type fspmm_dispatch (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FCat, MZO)`
- `template<class Field , class SM , class FCat , class MZO >`  
`std::enable_if< std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineFloat >::value) | std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineIntTag >::value >::type fspmm_dispatch (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FCat, MZO)`
- `template<class Field , class SM >`  
`void fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::GenericTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< support_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::UnparametricTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< !support_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::UnparametricTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< support_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::ModularTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< !support_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::ModularTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`void fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::GenericTag, ZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< support_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::UnparametricTag, ZOSparseMatrix)`



- `template<class Field , class SM >`  
`std::enable_if<!support\_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::UnparametricTag, ZOSparseMatrix)`
- `template<class Field , class SM >`  
`void fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::ModularTag, ZOSparseMatrix)`
- `template<class Field , class SM , class FCat , class MZO >`  
`std::enable_if<!(std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineFloat >::value)|std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineIntTag >::value)>::type pfspmm\_dispatch (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FCat, MZO)`
- `template<class Field , class SM , class FCat , class MZO >`  
`std::enable_if< std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineFloat >::value)|std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineIntTag >::value >::type pfspmm\_dispatch (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FCat, MZO)`
- `template<class Field , class SM >`  
`void pfspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::GenericTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< support\_simd< typenameField::Element >::value >::type pfspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::UnparametricTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if<!support\_simd< typenameField::Element >::value >::type pfspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::UnparametricTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< support\_simd< typenameField::Element >::value >::type pfspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::ModularTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if<!support\_simd< typenameField::Element >::value >::type pfspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::ModularTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`void pfspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::GenericTag, ZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< support\_simd< typenameField::Element >::value >::type pfspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::UnparametricTag, ZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if<!support\_simd< typenameField::Element >::value >::type pfspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::UnparametricTag, ZOSparseMatrix)`
- `template<class Field , class SM >`  
`void pfspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement\_ptr x, int ldx, typename Field::Element\_ptr y, int ldy, FieldCategories::ModularTag, ZOSparseMatrix)`
- `template<class Field , class SM >`  
`void pfspmv (const Field &F, const SM &A, typename Field::ConstElement\_ptr x, typename Field::Element\_ptr y, FieldCategories::GenericTag, std::false_type)`
- `template<class Field , class SM >`  
`void pfspmv (const Field &F, const SM &A, typename Field::ConstElement\_ptr x, typename Field::Element\_ptr y, FieldCategories::UnparametricTag, std::false_type)`

- `template<class Field, class SM >`  
`void pfspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::ModularTag, std::false_type)`
- `template<class Field, class SM >`  
`void pfspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::GenericTag, std::true_type)`
- `template<class Field, class SM >`  
`void pfspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag, std::true_type)`
- `template<class Field, class SM >`  
`void pfspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::ModularTag, std::true_type)`
- `template<class Field, class SM >`  
`void fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, const typename Field::Element &beta, typename Field::Element_ptr y)`
- `template<class Field, class SM >`  
`void fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, const typename Field::Element &beta, typename Field::Element_ptr y, int ldy)`

### 13.137.1 Macro Definition Documentation

#### 13.137.1.1 `index_t`

```
#define index_t uint32_t
```

#### 13.137.1.2 `ROUND_DOWN`

```
#define ROUND_DOWN(
 x,
 s) ((x) & ~((s)-1))
```

#### 13.137.1.3 `__FFLASFFPACK_CACHE_LINE_SIZE`

```
#define __FFLASFFPACK_CACHE_LINE_SIZE 64
```

#### 13.137.1.4 `assume_aligned`

```
#define assume_aligned(
 pout,
 pin,
 v) decltype(pin) pout = pin;
```

#### 13.137.1.5 `DENSE_THRESHOLD`

```
#define DENSE_THRESHOLD 0.5
```

## 13.138 `fflas_sparse.inl` File Reference

### Namespaces

- namespace `FFLAS`
- namespace `FFLAS::sparse_details`

### Macros

- `#define __FFLASFFPACK_fflas_fflas_sparse_INL`

## Functions

- `template<class Field >`  
`void init_y (const Field &F, const size_t m, const typename Field::Element b, typename Field::Element_ptr y)`
- `template<class Field >`  
`void init_y (const Field &F, const size_t m, const size_t n, const typename Field::Element b, typename Field::Element_ptr y, const int ldy)`
- `template<class Field , class SM , class FC , class MZO >`  
`std::enable_if<!(std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineFloat >::value)||std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineIntTag >::value)>::type fspmv_dispatch (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FC fc, MZO mzo)`
- `template<class Field , class SM , class FC , class MZO >`  
`std::enable_if< std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineFloat >::value)||std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineIntTag >::value >::type fspmv_dispatch (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FC fc, MZO mzo)`
- `template<class Field , class SM >`  
`void fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::GenericTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if<lisSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< isSparseMatrixSimdFormat< Field, SM >::value &&support_simd< typenameField::Element >::value >::type fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if<lisSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::ModularTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< isSparseMatrixSimdFormat< Field, SM >::value &&support_simd< typenameField::Element >::value >::type fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::ModularTag, NotZOSparseMatrix)`
- `template<class Field , class SM >`  
`void fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::GenericTag, ZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if<lisSparseMatrixSimdFormat< Field, SM >::value >::type fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag, ZOSparseMatrix)`
- `template<class Field , class SM >`  
`std::enable_if< isSparseMatrixSimdFormat< Field, SM >::value &&support_simd< typenameField::Element >::value >::type fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag, ZOSparseMatrix)`
- `template<class Field , class SM >`  
`void fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::ModularTag, std::true_type)`
- `template<class Field , class SM , class FCat , class MZO >`  
`std::enable_if<!(std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineFloat >::value)||std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineIntTag >::value)>::type fspmm_dispatch (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FCat, MZO)`
- `template<class Field , class SM , class FCat , class MZO >`  
`std::enable_if< std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineFloat >::value)||std::is_same< typenameElementTraits< typenameField::Element >::value, ElementCategories::MachineIntTag >::value >::type fspmm_dispatch (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FCat, MZO)`

```
>::value >::type fspmm_dispatch (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr
x, int ldx, typename Field::Element_ptr y, int ldy, FCat, MZO)
```

- `template<class Field, class SM >`  
`void fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx,`  
`typename Field::Element_ptr y, int ldy, FieldCategories::GenericTag, NotZOSparseMatrix)`
- `template<class Field, class SM >`  
`std::enable_if< support_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM`  
`&A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy,`  
`FieldCategories::UnparametricTag, NotZOSparseMatrix)`
- `template<class Field, class SM >`  
`std::enable_if<!support_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM`  
`&A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy,`  
`FieldCategories::UnparametricTag, NotZOSparseMatrix)`
- `template<class Field, class SM >`  
`std::enable_if< support_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM`  
`&A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy,`  
`FieldCategories::ModularTag, NotZOSparseMatrix)`
- `template<class Field, class SM >`  
`std::enable_if<!support_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM`  
`&A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy,`  
`FieldCategories::ModularTag, NotZOSparseMatrix)`
- `template<class Field, class SM >`  
`void fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx,`  
`typename Field::Element_ptr y, int ldy, FieldCategories::GenericTag, ZOSparseMatrix)`
- `template<class Field, class SM >`  
`std::enable_if< support_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM`  
`&A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy,`  
`FieldCategories::UnparametricTag, ZOSparseMatrix)`
- `template<class Field, class SM >`  
`std::enable_if<!support_simd< typenameField::Element >::value >::type fspmm (const Field &F, const SM`  
`&A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy,`  
`FieldCategories::UnparametricTag, ZOSparseMatrix)`
- `template<class Field, class SM >`  
`void fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx,`  
`typename Field::Element_ptr y, int ldy, FieldCategories::ModularTag, ZOSparseMatrix)`
- `template<class Field, class SM >`  
`void fspmv (const Field &F, const SM &A, typename Field::ConstElement_ptr x, const typename`  
`Field::Element &beta, typename Field::Element_ptr y)`
- `template<class Field, class SM >`  
`void fspmm (const Field &F, const SM &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx,`  
`const typename Field::Element &beta, typename Field::Element_ptr y, int ldy)`

## 13.138.1 Macro Definition Documentation

### 13.138.1.1 \_\_FFLASFFPACK\_fflas\_fflas\_sparse\_INL

```
#define __FFLASFFPACK_fflas_fflas_sparse_INL
```

## 13.139 coo.h File Reference

```
#include "fflas-ffpack/fflas/fflas_sparse/coo/coo_utils.inl"
#include "fflas-ffpack/fflas/fflas_sparse/coo/coo_spmv.inl"
#include "fflas-ffpack/fflas/fflas_sparse/coo/coo_spmmm.inl"
```

## Data Structures

- struct [Sparse< \\_Field, SparseMatrix\\_t::COO >](#)
- struct [Sparse< \\_Field, SparseMatrix\\_t::COO\\_ZO >](#)

## Namespaces

- namespace [FFLAS](#)

## Functions

- template<class [Field](#) , class IndexT >  
void [sparse\\_init](#) (const [Field](#) &F, [Sparse< Field, SparseMatrix\\_t::COO >](#) &A, const IndexT \*row, const IndexT \*col, typename [Field::ConstElement\\_ptr](#) dat, uint64\_t rowdim, uint64\_t coldim, uint64\_t nnz)
- template<class [Field](#) , class IndexT >  
void [sparse\\_init](#) (const [Field](#) &F, [Sparse< Field, SparseMatrix\\_t::COO\\_ZO >](#) &A, const IndexT \*row, const IndexT \*col, typename [Field::ConstElement\\_ptr](#) dat, uint64\_t rowdim, uint64\_t coldim, uint64\_t nnz)
- template<class [Field](#) >  
void [sparse\\_delete](#) (const [Sparse< Field, SparseMatrix\\_t::COO >](#) &A)
- template<class [Field](#) >  
void [sparse\\_delete](#) (const [Sparse< Field, SparseMatrix\\_t::COO\\_ZO >](#) &A)

# 13.140 coo\_spmv.inl File Reference

## Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details\\_impl](#)

## Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_sparse\\_coo\\_spmv\\_INL](#)

## Functions

- template<class [Field](#) >  
void [fspmv](#) (const [Field](#) &F, const [Sparse< Field, SparseMatrix\\_t::COO >](#) &A, size\_t blockSize, typename [Field::ConstElement\\_ptr](#) x\_, int ldx, typename [Field::Element\\_ptr](#) y\_, int ldy, [FieldCategories::GenericTag](#))
- template<class [Field](#) >  
void [fspmv](#) (const [Field](#) &F, const [Sparse< Field, SparseMatrix\\_t::COO >](#) &A, size\_t blockSize, typename [Field::ConstElement\\_ptr](#) x\_, int ldx, typename [Field::Element\\_ptr](#) y\_, int ldy, [FieldCategories::UnparametricTag](#))
- template<class [Field](#) >  
void [fspmv](#) (const [Field](#) &F, const [Sparse< Field, SparseMatrix\\_t::COO >](#) &A, size\_t blockSize, typename [Field::ConstElement\\_ptr](#) x\_, int ldx, typename [Field::Element\\_ptr](#) y\_, int ldy, const int64\_t kmax)
- template<class [Field](#) >  
void [fspmv\\_simd\\_aligned](#) (const [Field](#) &F, const [Sparse< Field, SparseMatrix\\_t::COO >](#) &A, size\_t blockSize, typename [Field::ConstElement\\_ptr](#) x\_, int ldx, typename [Field::Element\\_ptr](#) y\_, int ldy, const int64\_t kmax)
- template<class [Field](#) >  
void [fspmv\\_simd\\_unaligned](#) (const [Field](#) &F, const [Sparse< Field, SparseMatrix\\_t::COO >](#) &A, size\_t blockSize, typename [Field::ConstElement\\_ptr](#) x\_, int ldx, typename [Field::Element\\_ptr](#) y\_, int ldy, const int64\_t kmax)
- template<class [Field](#) >  
void [fspmv\\_one](#) (const [Field](#) &F, const [Sparse< Field, SparseMatrix\\_t::COO\\_ZO >](#) &A, size\_t blockSize, typename [Field::ConstElement\\_ptr](#) x\_, int ldx, typename [Field::Element\\_ptr](#) y\_, int ldy, [FieldCategories::GenericTag](#))

- `template<class Field >`  
`void fspmm_mone (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmm_one_simd_aligned (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_one_simd_unaligned (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_mone_simd_aligned (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_mone_simd_unaligned (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`

### 13.140.1 Macro Definition Documentation

#### 13.140.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_coo\_spmv\_INL

```
#define __FFLASFFPACK_fflas_sparse_coo_spmv_INL
```

## 13.141 coo\_spmv.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details\\_impl](#)

### Macros

- `#define __FFLASFFPACK_fflas_sparse_coo_spmv_INL`

### Functions

- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::COO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::COO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::COO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const uint64_t kmax)`
- `template<class Field >`  
`void fspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`

- `template<class Field >`  
`void fspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::COO_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`

### 13.141.1 Macro Definition Documentation

#### 13.141.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_coo\_spmv\_INL

```
#define __FFLASFFPACK_fflas_sparse_coo_spmv_INL
```

## 13.142 coo\_utils.inl File Reference

### Namespaces

- namespace [FFLAS](#)

### Macros

- `#define __FFLASFFPACK_fflas_sparse_coo_utils_INL`

### Functions

- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::COO > &A)`
- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::COO_ZO > &A)`
- `template<class Field, class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::COO > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class Field, class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::COO_ZO > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`

### 13.142.1 Macro Definition Documentation

#### 13.142.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_coo\_utils\_INL

```
#define __FFLASFFPACK_fflas_sparse_coo_utils_INL
```

## 13.143 csr.h File Reference

```
#include "fflas-ffpack/fflas/fflas_sparse/csr/csr_utils.inl"
#include "fflas-ffpack/fflas/fflas_sparse/csr/csr_spmv.inl"
#include "fflas-ffpack/fflas/fflas_sparse/csr/csr_spmv.inl"
```

### Data Structures

- struct `Sparse< _Field, SparseMatrix_t::CSR >`
- struct `Sparse< _Field, SparseMatrix_t::CSR_ZO >`

### Namespaces

- namespace [FFLAS](#)



## Functions

- `template<class Field, class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::CSR > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class Field, class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::CSR_ZO > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::CSR > &A)`
- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::CSR_ZO > &A)`

## 13.144 csr\_pspmm.inl File Reference

### Namespaces

- namespace `FFLAS`
- namespace `FFLAS::sparse_details_impl`

### Macros

- `#define __FFLASFFPACK_fflas_sparse_CSR_pspmm_INL`

### Functions

- `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, const int64_t kmax)`
- `template<class Field >`  
`void pfspmm_one (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmm_mone (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmm_one (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmm_mone (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`

### 13.144.1 Macro Definition Documentation

#### 13.144.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_pspmm\_INL

```
#define __FFLASFFPACK_fflas_sparse_CSR_pspmm_INL
```



## 13.145 csr\_pspmv.inl File Reference

```
#include <thread>
```

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details\\_impl](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_sparse\\_CSR\\_pspmv\\_INL](#)

### Functions

- template<class [Field](#) >  
void [pfspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::GenericTag](#))
- template<class [Field](#) >  
void [pfspmv\\_task](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, const [index\\_t](#) iStart, const [index\\_t](#) iStop, [FieldCategories::UnparametricTag](#))
- template<class [Field](#) >  
void [pfspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::UnparametricTag](#))
- template<class [Field](#) >  
void [pfspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, const [int64\\_t](#) kmax)
- template<class [Field](#) >  
void [pfspmv\\_one](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::GenericTag](#))
- template<class [Field](#) >  
void [pfspmv\\_mone](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::GenericTag](#))
- template<class [Field](#) >  
void [pfspmv\\_one](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::UnparametricTag](#))
- template<class [Field](#) >  
void [pfspmv\\_mone](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::UnparametricTag](#))

### 13.145.1 Macro Definition Documentation

#### 13.145.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_pspmv\_INL

```
#define __FFLASFFPACK_fflas_sparse_CSR_pspmv_INL
```

## 13.146 csr\_spmv.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details\\_impl](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_sparse\\_CSR\\_spmv\\_INL](#)

## Functions

- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, index_t blockSize, typename Field::ConstElement_ptr x_, index_t ldx, typename Field::Element_ptr y_, index_t ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_simd_aligned (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_simd_unaligned (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, const int64_t kmax)`
- `template<class Field >`  
`void fspmm_one (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmm_mone (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmm_one_simd_aligned (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_one_simd_unaligned (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_mone_simd_aligned (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_mone_simd_unaligned (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`

## 13.146.1 Macro Definition Documentation

### 13.146.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_spmv\_INL

```
#define __FFLASFFPACK_fflas_sparse_CSR_spmv_INL
```

## 13.147 csr\_spmv.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details\\_impl](#)

## Macros

- `#define __FFLASFFPACK_fflas_sparse_CSR_spmv_INL`

## Functions

- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const int64_t kmax)`
- `template<class Field >`  
`void fspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`

### 13.147.1 Macro Definition Documentation

#### 13.147.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_spmv\_INL

```
#define __FFLASFFPACK_fflas_sparse_CSR_spmv_INL
```

## 13.148 csr\_utils.inl File Reference

### Namespaces

- namespace `FFLAS`

### Functions

- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::CSR > &A)`
- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::CSR_ZO > &A)`
- `template<class Field >`  
`std::ostream & sparse_print (std::ostream &os, const Sparse< Field, SparseMatrix_t::CSR > &A)`
- `template<class IndexT >`  
`void sparse_init (const Givaro::Modular< Givaro::Integer > &F, Sparse< Givaro::Modular< Givaro::Integer >, SparseMatrix_t::CSR > &A, const IndexT *row, const IndexT *col, Givaro::Integer *dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class IndexT >`  
`void sparse_init (const Givaro::ZRing< Givaro::Integer > &F, Sparse< Givaro::ZRing< Givaro::Integer >, SparseMatrix_t::CSR_ZO > &A, const IndexT *row, const IndexT *col, Givaro::Integer *dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`

- `template<class IndexT, size_t RECINT_SIZE>`  
`void sparse_init (const Givaro::ZRing< RecInt::rmint< RECINT_SIZE > > &F, Sparse< Givaro::ZRing< RecInt::rmint< RECINT_SIZE > >, SparseMatrix_t::CSR_ZO > &A, const IndexT *row, const IndexT *col, typename Givaro::ZRing< RecInt::rmint< RECINT_SIZE > >::Element_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class IndexT, size_t RECINT_SIZE>`  
`void sparse_init (const Givaro::ZRing< RecInt::rmint< RECINT_SIZE > > &F, Sparse< Givaro::ZRing< RecInt::rmint< RECINT_SIZE > >, SparseMatrix_t::CSR > &A, const IndexT *row, const IndexT *col, typename Givaro::ZRing< RecInt::rmint< RECINT_SIZE > >::Element_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class Field, class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::CSR > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class Field, class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::CSR_ZO > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`

## 13.149 csr\_hyb.h File Reference

```
#include "fflas-ffpack/fflas/fflas_sparse/csr_hyb/csr_hyb_utils.inl"
#include "fflas-ffpack/fflas/fflas_sparse/csr_hyb/csr_hyb_spmv.inl"
#include "fflas-ffpack/fflas/fflas_sparse/csr_hyb/csr_hyb_spmmm.inl"
```

### Data Structures

- struct `Sparse< _Field, SparseMatrix_t::CSR_HYB >`

### Namespaces

- namespace `FFLAS`

### Functions

- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::CSR_HYB > &A)`
- `template<class Field, class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::CSR_HYB > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`

## 13.150 csr\_hyb\_pspmm.inl File Reference

### Namespaces

- namespace `FFLAS`
- namespace `FFLAS::sparse_details_impl`

### Macros

- `#define __FFLASFFPACK_fflas_sparse_CSR_HYB_pspmm_INL`

### Functions

- `template<class Field >`  
`void pfpmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, size_t blockSize, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::GenericTag)`

- `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, size_t blockSize, type-`  
`name Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, size_t blockSize, type-`  
`name Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, size_t blockSize, type-`  
`name Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, size_t blockSize, type-`  
`name Field::ConstElement_ptr x, typename Field::Element_ptr y, const int64_t kmax)`
- `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, size_t blockSize, type-`  
`name Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, const int64_t kmax)`

### 13.150.1 Macro Definition Documentation

#### 13.150.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_pspmm\_INL

```
#define __FFLASFFPACK_fflas_sparse_CSR_HYB_pspmm_INL
```

## 13.151 csr\_hyb\_pspmv.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details\\_impl](#)

### Macros

- `#define` [\\_\\_FFLASFFPACK\\_fflas\\_sparse\\_CSR\\_HYB\\_pspmv\\_INL](#)

### Functions

- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, typename`  
`Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, typename`  
`Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, typename`  
`Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const int64_t kmax)`

### 13.151.1 Macro Definition Documentation

#### 13.151.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_pspmv\_INL

```
#define __FFLASFFPACK_fflas_sparse_CSR_HYB_pspmv_INL
```

## 13.152 csr\_hyb\_spmv.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details\\_impl](#)

## Macros

- `#define __FFLASFFPACK_fflas_sparse_CSR_HYB_spmv_INL`

## Functions

- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, size_t blockSize, type-`  
`name Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, size_t blockSize, type-`  
`name Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, size_t blockSize, type-`  
`name Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, const int64_t kmax)`

### 13.152.1 Macro Definition Documentation

#### 13.152.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_spmv\_INL

```
#define __FFLASFFPACK_fflas_sparse_CSR_HYB_spmv_INL
```

## 13.153 csr\_hyb\_spmv.inl File Reference

### Namespaces

- namespace `FFLAS`
- namespace `FFLAS::sparse_details_impl`

### Macros

- `#define __FFLASFFPACK_fflas_sparse_CSR_HYB_spmv_INL`

### Functions

- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, typename`  
`Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, typename`  
`Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::CSR_HYB > &A, typename`  
`Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const uint64_t kmax)`

### 13.153.1 Macro Definition Documentation

#### 13.153.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_spmv\_INL

```
#define __FFLASFFPACK_fflas_sparse_CSR_HYB_spmv_INL
```

## 13.154 csr\_hyb\_utils.inl File Reference

### Data Structures

- struct `Info`
- struct `Coo< ValT, ldxT >`

## Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::csr\\_hyb\\_details](#)

## Macros

- `#define` [\\_\\_FFLASFFPACK\\_fflas\\_sparse\\_CSR\\_HYB\\_utils\\_INL](#)

## Functions

- `template<class Field >`  
void [sparse\\_delete](#) (const [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_HYB](#) > &A)
- `template<class Field , class IndexT >`  
void [sparse\\_init](#) (const [Field](#) &F, [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_HYB](#) > &A, const [IndexT](#) \*row, const [IndexT](#) \*col, typename [Field::ConstElement\\_ptr](#) dat, [uint64\\_t](#) rowdim, [uint64\\_t](#) coldim, [uint64\\_t](#) nnz)

## 13.154.1 Macro Definition Documentation

### 13.154.1.1 [\\_\\_FFLASFFPACK\\_fflas\\_sparse\\_CSR\\_HYB\\_utils\\_INL](#)

```
#define __FFLASFFPACK_fflas_sparse_CSR_HYB_utils_INL
```

## 13.155 ell.h File Reference

```
#include "fflas-ffpack/fflas/fflas_sparse/ell/ell_utils.inl"
#include "fflas-ffpack/fflas/fflas_sparse/ell/ell_spmv.inl"
#include "fflas-ffpack/fflas/fflas_sparse/ell/ell_spmmm.inl"
```

## Data Structures

- struct [Sparse](#)< [\\_Field](#), [SparseMatrix\\_t::ELL](#) >
- struct [Sparse](#)< [\\_Field](#), [SparseMatrix\\_t::ELL\\_ZO](#) >

## Namespaces

- namespace [FFLAS](#)

## Functions

- `template<class Field , class IndexT >`  
void [sparse\\_init](#) (const [Field](#) &F, [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL](#) > &A, const [IndexT](#) \*row, const [IndexT](#) \*col, typename [Field::ConstElement\\_ptr](#) dat, [uint64\\_t](#) rowdim, [uint64\\_t](#) coldim, [uint64\\_t](#) nnz)
- `template<class Field , class IndexT >`  
void [sparse\\_init](#) (const [Field](#) &F, [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_ZO](#) > &A, const [IndexT](#) \*row, const [IndexT](#) \*col, typename [Field::ConstElement\\_ptr](#) dat, [uint64\\_t](#) rowdim, [uint64\\_t](#) coldim, [uint64\\_t](#) nnz)
- `template<class Field >`  
void [sparse\\_delete](#) (const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL](#) > &A)
- `template<class Field >`  
void [sparse\\_delete](#) (const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_ZO](#) > &A)

## 13.156 ell\_pspmm.inl File Reference

## Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details\\_impl](#)

## Macros

- `#define __FFLASFFPACK_fflas_sparse_ELL_pspmm_INL`

## Functions

- `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, size_t blockSize, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, size_t blockSize, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmm (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, size_t blockSize, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, const int64_t kmax)`
- `template<class Field, class Func >`  
`void pfspmm_zo (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x, typename Field::Element_ptr y, Func &&func)`
- `template<class Field, class Func >`  
`void pfspmm_zo (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x, int ldx, typename Field::Element_ptr y, int ldy, Func &&func)`

## 13.156.1 Macro Definition Documentation

### 13.156.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_pspmm\_INL

```
#define __FFLASFFPACK_fflas_sparse_ELL_pspmm_INL
```

## 13.157 ell\_pspmv.inl File Reference

### Namespaces

- namespace `FFLAS`
- namespace `FFLAS::sparse_details_impl`

### Macros

- `#define __FFLASFFPACK_fflas_sparse_ELL_pspmv_INL`

### Functions

- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const int64_t kmax)`



- `template<class Field >`  
`void pfspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`

### 13.157.1 Macro Definition Documentation

#### 13.157.1.1 \_\_FflasFfpack\_fflas\_sparse\_ELL\_pspmv\_INL

```
#define __FflasFfpack_fflas_sparse_ELL_pspmv_INL
```

## 13.158 ell\_spmv.inl File Reference

### Namespaces

- namespace `FFLAS`
- namespace `FFLAS::sparse_details_impl`

### Macros

- `#define __FflasFfpack_fflas_sparse_ELL_spmv_INL`

### Functions

- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, const int64_t kmax)`
- `template<class Field >`  
`void fspmm_mone (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmm_one (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmm_mone (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_one (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy, FieldCategories::UnparametricTag)`

- `template<class Field >`  
`void fspmm_one_simd_aligned (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A,`  
`size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy,`  
`FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_one_simd_unaligned (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A,`  
`size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy,`  
`FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_mone_simd_aligned (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A,`  
`size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy,`  
`FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmm_mone_simd_unaligned (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A,`  
`size_t blockSize, typename Field::ConstElement_ptr x_, int ldx, typename Field::Element_ptr y_, int ldy,`  
`FieldCategories::UnparametricTag)`

### 13.158.1 Macro Definition Documentation

#### 13.158.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_spmv\_INL

```
#define __FFLASFFPACK_fflas_sparse_ELL_spmv_INL
```

## 13.159 ell\_spmv.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details\\_impl](#)

### Macros

- `#define` [\\_\\_FFLASFFPACK\\_fflas\\_sparse\\_ELL\\_spmv\\_INL](#)

### Functions

- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, typename Field::ConstElement_ptr`  
`x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, typename Field::ConstElement_ptr`  
`x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL > &A, typename Field::ConstElement_ptr`  
`x_, typename Field::Element_ptr y_, const uint64_t kmax)`
- `template<class Field >`  
`void fspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, typename`  
`Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, typename`  
`Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, typename`  
`Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_ZO > &A, typename`  
`Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`

## 13.159.1 Macro Definition Documentation

### 13.159.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_spmv\_INL

```
#define __FFLASFFPACK_fflas_sparse_ELL_spmv_INL
```

## 13.160 ell\_utils.inl File Reference

```
#include <vector>
```

### Namespaces

- namespace [FFLAS](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_sparse\\_ELL\\_utils\\_INL](#)

### Functions

- template<class [Field](#) >  
void [sparse\\_delete](#) (const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL](#) > &A)
- template<class [Field](#) >  
void [sparse\\_delete](#) (const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_ZO](#) > &A)
- template<class [Field](#) , class [IndexT](#) >  
void [sparse\\_init](#) (const [Field](#) &F, [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL](#) > &A, const [IndexT](#) \*row, const [IndexT](#) \*col, typename [Field::ConstElement\\_ptr](#) dat, uint64\_t rowdim, uint64\_t coldim, uint64\_t nnz)
- template<class [Field](#) , class [IndexT](#) >  
void [sparse\\_init](#) (const [Field](#) &F, [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_ZO](#) > &A, const [IndexT](#) \*row, const [IndexT](#) \*col, typename [Field::ConstElement\\_ptr](#) dat, uint64\_t rowdim, uint64\_t coldim, uint64\_t nnz)

## 13.160.1 Macro Definition Documentation

### 13.160.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_utils\_INL

```
#define __FFLASFFPACK_fflas_sparse_ELL_utils_INL
```

## 13.161 ell\_simd.h File Reference

```
#include "fflas-ffpack/fflas/fflas_sparse/ell_simd/ell_simd_utils.inl"
#include "fflas-ffpack/fflas/fflas_sparse/ell_simd/ell_simd_spmv.inl"
```

### Data Structures

- struct [Sparse](#)< [\\_Field](#), [SparseMatrix\\_t::ELL\\_simd](#) >
- struct [Sparse](#)< [\\_Field](#), [SparseMatrix\\_t::ELL\\_simd\\_ZO](#) >

### Namespaces

- namespace [FFLAS](#)

### Functions

- template<class [Field](#) , class [IndexT](#) >  
void [sparse\\_init](#) (const [Field](#) &F, [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_simd](#) > &A, const [IndexT](#) \*row, const [IndexT](#) \*col, typename [Field::ConstElement\\_ptr](#) dat, uint64\_t rowdim, uint64\_t coldim, uint64\_t nnz)

- `template<class Field, class IndexT >`  
`void sparse_init (const Field &F, Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A, const IndexT *row,`  
`const IndexT *col, typename Field::ConstElement_ptr dat, uint64_t rowdim, uint64_t coldim, uint64_t nnz)`
- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::ELL_simd > &A)`
- `template<class Field >`  
`void sparse_delete (const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A)`

## 13.162 ell\_simd\_pspmv.inl File Reference

### Namespaces

- namespace `FFLAS`
- namespace `FFLAS::sparse_details_impl`

### Macros

- `#define __FFLASFFPACK_fflas_sparse_ELL_simd_pspmv_INL`

### Functions

- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd > &A, typename`  
`Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd > &A, typename`  
`Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd > &A, typename`  
`Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const uint64_t kmax)`
- `template<class Field >`  
`void pfspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A, typename`  
`Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A, typename`  
`Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void pfspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A, typename`  
`Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void pfspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A, typename`  
`Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`

### 13.162.1 Macro Definition Documentation

#### 13.162.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_simd\_pspmv\_INL

```
#define __FFLASFFPACK_fflas_sparse_ELL_simd_pspmv_INL
```

## 13.163 ell\_simd\_spmv.inl File Reference

### Namespaces

- namespace `FFLAS`
- namespace `FFLAS::sparse_details_impl`

**Macros**

- `#define __FFLASFFPACK_fflas_sparse_ELL_simd_spmv_INL`

**Functions**

- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv_simd (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_simd (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const uint64_t kmax)`
- `template<class Field >`  
`void fspmv (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, const uint64_t kmax)`
- `template<class Field >`  
`void fspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::GenericTag)`
- `template<class Field >`  
`void fspmv_one (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_mone (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_one_simd (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`
- `template<class Field >`  
`void fspmv_mone_simd (const Field &F, const Sparse< Field, SparseMatrix_t::ELL_simd_ZO > &A, typename Field::ConstElement_ptr x_, typename Field::Element_ptr y_, FieldCategories::UnparametricTag)`

**13.163.1 Macro Definition Documentation****13.163.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_simd\_spmv\_INL**

```
#define __FFLASFFPACK_fflas_sparse_ELL_simd_spmv_INL
```

**13.164 ell\_simd\_utils.inl File Reference****Namespaces**

- namespace `FFLAS`

**Macros**

- `#define __FFLASFFPACK_fflas_sparse_ELL_simd_utils_INL`

## Functions

- template<class [Field](#) >  
void [sparse\\_delete](#) (const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_simd](#) > &A)
- template<class [Field](#) >  
void [sparse\\_delete](#) (const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_simd\\_ZO](#) > &A)
- template<class [Field](#) >  
void [sparse\\_print](#) (const [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_simd](#) > &A)
- template<class [Field](#) , class [IndexT](#) >  
void [sparse\\_init](#) (const [Field](#) &F, [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_simd](#) > &A, const [IndexT](#) \*row, const [IndexT](#) \*col, typename [Field::ConstElement\\_ptr](#) dat, [uint64\\_t](#) rowdim, [uint64\\_t](#) coldim, [uint64\\_t](#) nnz)
- template<class [Field](#) , class [IndexT](#) >  
void [sparse\\_init](#) (const [Field](#) &F, [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_simd\\_ZO](#) > &A, const [IndexT](#) \*row, const [IndexT](#) \*col, typename [Field::ConstElement\\_ptr](#) dat, [uint64\\_t](#) rowdim, [uint64\\_t](#) coldim, [uint64\\_t](#) nnz)

## 13.164.1 Macro Definition Documentation

### 13.164.1.1 [\\_\\_FFLASFFPACK\\_fflas\\_sparse\\_ELL\\_simd\\_utils\\_INL](#)

```
#define __FFLASFFPACK_fflas_sparse_ELL_simd_utils_INL
```

## 13.165 [hyb\\_zo.h](#) File Reference

```
#include "fflas-ffpack/fflas/fflas_sparse/hyb_zo/hyb_zo_utils.inl"
#include "fflas-ffpack/fflas/fflas_sparse/hyb_zo/hyb_zo_spmv.inl"
#include "fflas-ffpack/fflas/fflas_sparse/hyb_zo/hyb_zo_spm্ম.inl"
```

## Data Structures

- struct [Sparse](#)< [\\_Field](#), [SparseMatrix\\_t::HYB\\_ZO](#) >

## Namespaces

- namespace [FFLAS](#)

## 13.166 [hyb\\_zo\\_pspmm.inl](#) File Reference

## Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details\\_impl](#)

## Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_sparse\\_HYB\\_ZO\\_pspmm\\_INL](#)

## Functions

- template<class [Field](#) >  
void [pfsppmm](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::HYB\\_ZO](#) > &A, [size\\_t](#) blockSize, typename [Field::ConstElement\\_ptr](#) x, [int](#) ldx, typename [Field::Element\\_ptr](#) y, [int](#) ldy, [FieldCategories::GenericTag](#))
- template<class [Field](#) >  
void [pfsppmm](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::HYB\\_ZO](#) > &A, [size\\_t](#) blockSize, typename [Field::ConstElement\\_ptr](#) x, [int](#) ldx, typename [Field::Element\\_ptr](#) y, [int](#) ldy, [FieldCategories::UnparametricTag](#))
- template<class [Field](#) >  
void [pfsppmm](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::HYB\\_ZO](#) > &A, [size\\_t](#) blockSize, typename [Field::ConstElement\\_ptr](#) x, [int](#) ldx, typename [Field::Element\\_ptr](#) y, [int](#) ldy, [uint64\\_t](#) kmax)

## 13.166.1 Macro Definition Documentation

### 13.166.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_pspmm\_INL

```
#define __FFLASFFPACK_fflas_sparse_HYB_ZO_pspmm_INL
```

## 13.167 hyb\_zo\_pspmv.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details\\_impl](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_sparse\\_HYB\\_ZO\\_pspmv\\_INL](#)

### Functions

- template<class [Field](#) >  
void [pfspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::HYB\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x, typename [Field::Element\\_ptr](#) y, [FieldCategories::GenericTag](#))
- template<class [Field](#) >  
void [pfspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::HYB\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x, typename [Field::Element\\_ptr](#) y, [FieldCategories::UnparametricTag](#))
- template<class [Field](#) >  
void [pfspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::HYB\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x, typename [Field::Element\\_ptr](#) y, uint64\_t kmax)

## 13.167.1 Macro Definition Documentation

### 13.167.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_pspmv\_INL

```
#define __FFLASFFPACK_fflas_sparse_HYB_ZO_pspmv_INL
```

## 13.168 hyb\_zo\_spmv.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details\\_impl](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_sparse\\_HYB\\_ZO\\_spmv\\_INL](#)

### Functions

- template<class [Field](#) >  
void [fspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::HYB\\_ZO](#) > &A, size\_t blockSize, typename [Field::ConstElement\\_ptr](#) x, int ldx, typename [Field::Element\\_ptr](#) y, int ldy, [FieldCategories::GenericTag](#))
- template<class [Field](#) >  
void [fspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::HYB\\_ZO](#) > &A, size\_t blockSize, typename [Field::ConstElement\\_ptr](#) x, int ldx, typename [Field::Element\\_ptr](#) y, int ldy, [FieldCategories::UnparametricTag](#))
- template<class [Field](#) >  
void [fspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::HYB\\_ZO](#) > &A, size\_t blockSize, typename [Field::ConstElement\\_ptr](#) x, int ldx, typename [Field::Element\\_ptr](#) y, int ldy, uint64\_t kmax)

### 13.168.1 Macro Definition Documentation

#### 13.168.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_spmv\_INL

```
#define __FFLASFFPACK_fflas_sparse_HYB_ZO_spmv_INL
```

## 13.169 hyb\_zo\_spmv.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details\\_impl](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_sparse\\_HYB\\_ZO\\_spmv\\_INL](#)

### Functions

- template<class [Field](#) >  
void [fspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::HYB\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x, typename [Field::Element\\_ptr](#) y, [FieldCategories::GenericTag](#))
- template<class [Field](#) >  
void [fspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::HYB\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x, typename [Field::Element\\_ptr](#) y, [FieldCategories::UnparametricTag](#))
- template<class [Field](#) >  
void [fspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::HYB\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x, typename [Field::Element\\_ptr](#) y, [uint64\\_t](#) kmax)

### 13.169.1 Macro Definition Documentation

#### 13.169.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_spmv\_INL

```
#define __FFLASFFPACK_fflas_sparse_HYB_ZO_spmv_INL
```

## 13.170 hyb\_zo\_utils.inl File Reference

### Namespaces

- namespace [FFLAS](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_sparse\\_HYB\\_ZO\\_utils\\_INL](#)

### Functions

- template<class [Field](#) >  
void [sparse\\_delete](#) (const [Sparse](#)< [Field](#), [SparseMatrix\\_t::HYB\\_ZO](#) > &A)
- template<class [Field](#) , class [IndexT](#) >  
void [sparse\\_init](#) (const [Field](#) &F, [Sparse](#)< [Field](#), [SparseMatrix\\_t::HYB\\_ZO](#) > &A, const [IndexT](#) \*row, const [IndexT](#) \*col, typename [Field::ConstElement\\_ptr](#) dat, [uint64\\_t](#) rowdim, [uint64\\_t](#) coldim, [uint64\\_t](#) nnz)
- template<typename [\\_Field](#) >  
std::ostream & [operator<<](#) (std::ostream &os, const [Sparse](#)< [\\_Field](#), [SparseMatrix\\_t::HYB\\_ZO](#) > &A)

### 13.170.1 Macro Definition Documentation

#### 13.170.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_utils\_INL

```
#define __FFLASFFPACK_fflas_sparse_HYB_ZO_utils_INL
```



## 13.171 read\_sparse.h File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <fstream>
#include <string>
#include <cstdlib>
#include <iterator>
```

### Data Structures

- struct [Coo< Field >](#)
- struct [readMyMachineType< Field, T >](#)
- struct [readMyMachineType< Field, mpz\\_t >](#)

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::details\\_spmv](#)

### Macros

- [#define DNS\\_BIN\\_VER 0](#)
- [#define mask\\_t uint64\\_t](#)

### Functions

- [template<class Field, bool sorted = true, bool read\\_integer = false>  
void readSmsFormat \(const std::string &path, const Field &f, index\\_t \\*&row, index\\_t \\*&col, typename Field::Element\\_ptr &val, index\\_t &rowdim, index\\_t &coldim, uint64\\_t &n nz\)](#)
- [template<class Field >  
void readSprFormat \(const std::string &path, const Field &f, index\\_t \\*&row, index\\_t \\*&col, typename Field::Element\\_ptr &val, index\\_t &rowdim, index\\_t &coldim, uint64\\_t &n nz\)](#)
- [template<class T >  
std::enable\\_if< std::is\\_integral< T >::value, int > getDataType \(\)](#)
- [template<class T >  
std::enable\\_if< std::is\\_floating\\_point< T >::value, int > getDataType \(\)](#)
- [template<class T >  
std::enable\\_if< std::is\\_same< T, mpz\\_t >::value, int > getDataType \(\)](#)
- [template<class T >  
int getDataType \(\)](#)
- [template<class Field >  
void readMachineType \(const Field &F, typename Field::Element &modulo, typename Field::Element\\_ptr val, std::ifstream &file, const uint64\\_t dims, const mask\\_t data\\_type, const mask\\_t field\\_desc\)](#)
- [template<class Field >  
void readDnsFormat \(const std::string &path, const Field &F, index\\_t &rowdim, index\\_t &coldim, typename Field::Element\\_ptr &val\)](#)
- [template<class Field >  
void writeDnsFormat \(const std::string &path, const Field &F, const index\\_t &rowdim, const index\\_t &coldim, typename Field::Element\\_ptr A, index\\_t ldA\)](#)

### 13.171.1 Macro Definition Documentation

#### 13.171.1.1 DNS\_BIN\_VER

```
#define DNS_BIN_VER 0
```

### 13.171.1.2 mask\_t

```
#define mask_t uint64_t
```

## 13.172 sell.h File Reference

```
#include "fflas-ffpack/fflas/fflas_sparse/sell/sell_utils.inl"
#include "fflas-ffpack/fflas/fflas_sparse/sell/sell_spmv.inl"
```

### Data Structures

- struct [Sparse<\\_Field, SparseMatrix\\_t::SELL>](#)
- struct [Sparse<\\_Field, SparseMatrix\\_t::SELL\\_ZO>](#)

### Namespaces

- namespace [FFLAS](#)

## 13.173 sell\_pspmv.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details\\_impl](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_fflas\\_sparse\\_sell\\_pspmv\\_INL](#)

### Functions

- template<class [Field](#)>  
void [pfspmv](#) (const [Field](#) &F, const [Sparse<Field, SparseMatrix\\_t::SELL>](#) &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::GenericTag](#))
- template<class [Field](#)>  
void [pfspmv](#) (const [Field](#) &F, const [Sparse<Field, SparseMatrix\\_t::SELL>](#) &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::UnparametricTag](#))
- template<class [Field](#)>  
void [pfspmv](#) (const [Field](#) &F, const [Sparse<Field, SparseMatrix\\_t::SELL>](#) &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, const int64\_t kmax)
- template<class [Field](#)>  
void [pfspmv\\_one](#) (const [Field](#) &F, const [Sparse<Field, SparseMatrix\\_t::SELL\\_ZO>](#) &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::GenericTag](#))
- template<class [Field](#)>  
void [pfspmv\\_mone](#) (const [Field](#) &F, const [Sparse<Field, SparseMatrix\\_t::SELL\\_ZO>](#) &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::GenericTag](#))
- template<class [Field](#)>  
void [pfspmv\\_one](#) (const [Field](#) &F, const [Sparse<Field, SparseMatrix\\_t::SELL\\_ZO>](#) &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::UnparametricTag](#))
- template<class [Field](#)>  
void [pfspmv\\_mone](#) (const [Field](#) &F, const [Sparse<Field, SparseMatrix\\_t::SELL\\_ZO>](#) &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::UnparametricTag](#))

## 13.173.1 Macro Definition Documentation

### 13.173.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_sell\_pspmv\_INL

```
#define __FFLASFFPACK_fflas_sparse_sell_pspmv_INL
```

## 13.174 sell\_spmv.inl File Reference

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sparse\\_details\\_impl](#)

### Macros

- `#define` [\\_\\_FFLASFFPACK\\_fflas\\_sparse\\_sell\\_spmv\\_INL](#)

### Functions

- `template<class Field >`  
void [fspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::SELL](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::GenericTag](#))
- `template<class Field >`  
void [fspmv\\_simd](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::SELL](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::UnparametricTag](#))
- `template<class Field >`  
void [fspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::SELL](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::UnparametricTag](#))
- `template<class Field >`  
void [fspmv\\_simd](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::SELL](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, const [uint64\\_t](#) kmax)
- `template<class Field >`  
void [fspmv](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::SELL](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, const [uint64\\_t](#) kmax)
- `template<class Field >`  
void [fspmv\\_one](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::SELL\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::GenericTag](#))
- `template<class Field >`  
void [fspmv\\_mone](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::SELL\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::GenericTag](#))
- `template<class Field >`  
void [fspmv\\_one\\_simd](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::SELL\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::UnparametricTag](#))
- `template<class Field >`  
void [fspmv\\_mone\\_simd](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::SELL\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::UnparametricTag](#))
- `template<class Field >`  
void [fspmv\\_one](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::SELL\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::UnparametricTag](#))
- `template<class Field >`  
void [fspmv\\_mone](#) (const [Field](#) &F, const [Sparse](#)< [Field](#), [SparseMatrix\\_t::SELL\\_ZO](#) > &A, typename [Field::ConstElement\\_ptr](#) x\_, typename [Field::Element\\_ptr](#) y\_, [FieldCategories::UnparametricTag](#))

### 13.174.1 Macro Definition Documentation

#### 13.174.1.1 \_\_FFLASFFPACK\_fflas\_sparse\_sell\_spmv\_INL

```
#define __FFLASFFPACK_fflas_sparse_sell_spmv_INL
```

## 13.175 sell\_utils.inl File Reference

### Data Structures

- struct [Info](#)
- struct [Coo](#)< [ValT](#), [IdxT](#) >

## Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::sell\\_details](#)

## Macros

- `#define` [\\_\\_FFLASFFPACK\\_fflas\\_sparse\\_sell\\_utils\\_INL](#)

## Functions

- `template<class Field >`  
`void fspm(const Field &F, const Sparse< Field, SparseMatrix\_t::SELL\_ZO > &A, typename Field::ConstElement\_ptr x, typename Field::Element\_ptr y, FieldCategories::ModularTag)`
- `template<class Field >`  
`void sparse\_delete(const Sparse< Field, SparseMatrix\_t::SELL > &A)`
- `template<class Field >`  
`void sparse\_delete(const Sparse< Field, SparseMatrix\_t::SELL\_ZO > &A)`
- `template<class Field >`  
`void sparse\_print(const Sparse< Field, SparseMatrix\_t::SELL > &A)`
- `template<class Field, class IndexT >`  
`void sparse\_init(const Field &F, Sparse< Field, SparseMatrix\_t::SELL > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement\_ptr dat, uint64\_t rowdim, uint64\_t coldim, uint64\_t nnz, uint64\_t sigma=0)`
- `template<class Field, class IndexT >`  
`void sparse\_init(const Field &F, Sparse< Field, SparseMatrix\_t::SELL\_ZO > &A, const IndexT *row, const IndexT *col, typename Field::ConstElement\_ptr dat, uint64\_t rowdim, uint64\_t coldim, uint64\_t nnz)`

## 13.175.1 Macro Definition Documentation

### 13.175.1.1 [\\_\\_FFLASFFPACK\\_fflas\\_sparse\\_sell\\_utils\\_INL](#)

```
#define __FFLASFFPACK_fflas_sparse_sell_utils_INL
```

## 13.176 [sparse\\_matrix\\_traits.h](#) File Reference

```
#include <type_traits>
```

## Data Structures

- struct [isSparseMatrix](#)< [Field](#), [M](#) >
- struct [isSparseMatrix](#)< [Field](#), [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR](#) > >
- struct [isSparseMatrix](#)< [Field](#), [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_ZO](#) > >
- struct [isSparseMatrix](#)< [Field](#), [Sparse](#)< [Field](#), [SparseMatrix\\_t::COO](#) > >
- struct [isSparseMatrix](#)< [Field](#), [Sparse](#)< [Field](#), [SparseMatrix\\_t::COO\\_ZO](#) > >
- struct [isSparseMatrix](#)< [Field](#), [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL](#) > >
- struct [isSparseMatrix](#)< [Field](#), [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_ZO](#) > >
- struct [isSparseMatrix](#)< [Field](#), [Sparse](#)< [Field](#), [SparseMatrix\\_t::SELL](#) > >
- struct [isSparseMatrix](#)< [Field](#), [Sparse](#)< [Field](#), [SparseMatrix\\_t::SELL\\_ZO](#) > >
- struct [isSparseMatrix](#)< [Field](#), [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_simd](#) > >
- struct [isSparseMatrix](#)< [Field](#), [Sparse](#)< [Field](#), [SparseMatrix\\_t::ELL\\_simd\\_ZO](#) > >
- struct [isSparseMatrix](#)< [Field](#), [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_HYB](#) > >
- struct [isSparseMatrix](#)< [Field](#), [Sparse](#)< [Field](#), [SparseMatrix\\_t::HYB\\_ZO](#) > >
- struct [isZOSparseMatrix](#)< [F](#), [M](#) >
- struct [isZOSparseMatrix](#)< [Field](#), [Sparse](#)< [Field](#), [SparseMatrix\\_t::CSR\\_ZO](#) > >
- struct [isZOSparseMatrix](#)< [Field](#), [Sparse](#)< [Field](#), [SparseMatrix\\_t::COO\\_ZO](#) > >

- struct `isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_ZO > >`
- struct `isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::SELL_ZO > >`
- struct `isZOSparseMatrix< Field, Sparse< Field, SparseMatrix_t::ELL_simd_ZO > >`
- struct `isSparseMatrixSimdFormat< F, M >`
- struct `isSparseMatrixMKLFormat< F, M >`
- struct `tfn_plus`
- struct `tfn_mul`
- struct `tfn_mul_eq`
- struct `tfn_minus`
- struct `tfn_plus_eq`
- struct `tfn_minus_eq`
- struct `has_plus_impl< C >`
- struct `has_mul_impl< C >`
- struct `has_mul_eq_impl< C >`
- struct `has_plus_eq_impl< C >`
- struct `has_minus_eq_impl< C >`
- struct `has_minus_impl< C >`
- struct `has_operation< T >`

## Namespaces

- namespace `FFLAS`

## Typedefs

- using `ZOSparseMatrix` = `std::true_type`
- using `NotZOSparseMatrix` = `std::false_type`
- using `SimdSparseMatrix` = `std::true_type`
- using `NoSimdSparseMatrix` = `std::false_type`
- using `MKLSparseMatrixFormat` = `std::true_type`
- using `NotMKLSparseMatrixFormat` = `std::false_type`
- template<class T >  
using `has_plus` = `typename std::conditional< std::is_arithmetic< T >::value, std::true_type, has_plus_impl< T > >::type`
- template<class T >  
using `has_minus` = `typename std::conditional< std::is_arithmetic< T >::value, std::true_type, has_minus_impl< T > >::type`
- template<class T >  
using `has_equal` = `typename std::conditional< std::is_arithmetic< T >::value, std::true_type, std::is_copyable< T > & _assignable< T > >::type`
- template<class T >  
using `has_plus_eq` = `typename std::conditional< std::is_arithmetic< T >::value, std::true_type, has_plus_eq_impl< T > >::type`
- template<class T >  
using `has_minus_eq` = `typename std::conditional< std::is_arithmetic< T >::value, std::true_type, has_minus_eq_impl< T > >::type`
- template<class T >  
using `has_mul` = `typename std::conditional< std::is_arithmetic< T >::value, std::true_type, has_mul_impl< T > >::type`
- template<class T >  
using `has_mul_eq` = `typename std::conditional< std::is_arithmetic< T >::value, std::true_type, has_mul_eq_impl< T > >::type`

## 13.177 utils.h File Reference

```
#include <algorithm>
#include <numeric>
#include <vector>
```

### Data Structures

- struct [StatsMatrix](#)

### Namespaces

- namespace [FFLAS](#)

### Functions

- template<class It >  
double [computeDeviation](#) (It begin, It end)
- template<class Field >  
[StatsMatrix](#) [getStat](#) (const [Field](#) &F, const [index\\_t](#) \*row, const [index\\_t](#) \*col, typename [Field::ConstElement\\_ptr](#) val, uint64\_t rowdim, uint64\_t coldim, uint64\_t nnz)

## 13.178 fflas\_transpose.h File Reference

transpose the storage of the matrix (switch between row and col major mode)

```
#include "fflas-ffpack/utils/debug.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/fflas_memory.h"
#include "fflas-ffpack/fflas/fflas_simd.h"
```

### Data Structures

- struct [BlockTransposeSIMD](#)< [Field](#), [Simd](#), >

### Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::\\_ftranspose\\_impl](#)

### Macros

- #define [FFLAS\\_TRANSPOSE\\_BLOCKSIZE](#) 32
- #define [LD](#)(i) [R##i=Simd::loadu\(A+lda\\*i\)](#)
- #define [ST](#)(i) [Simd::storeu\(B+ldb\\*i,R##i\)](#)

### Functions

- template<size\_t bs, typename [Field](#) , typename [BTSimd](#) >  
void [not\\_inplace](#) (const [Field](#) &F, const [BTSimd](#) &BTS, const size\_t m, const size\_t n, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) B, const size\_t ldb)
- template<size\_t bs, typename [Field](#) , typename [BTSimd](#) >  
void [square\\_inplace](#) (const [Field](#) &F, const [BTSimd](#) &BTS, const size\_t m, typename [Field::Element\\_ptr](#) A, const size\_t lda)
- template<size\_t bs, typename [Field](#) , typename [BTSimd](#) >  
void [nonsquare\\_inplace\\_v1](#) (const [Field](#) &F, const [BTSimd](#) &BTS, const size\_t m, const size\_t n, typename [Field::Element\\_ptr](#) A)

- `template<size_t bs, typename Field, typename BTSimd >`  
`void nonsquare_inplace_v2 (const Field &F, const BTSimd &BTS, const size_t m, const size_t n, typename Field::Element_ptr A)`

### 13.178.1 Detailed Description

transpose the storage of the matrix (switch between row and col major mode)

### 13.178.2 Macro Definition Documentation

#### 13.178.2.1 FFLAS\_TRANSPOSE\_BLOCKSIZE

```
#define FFLAS_TRANSPOSE_BLOCKSIZE 32
```

#### 13.178.2.2 LD

```
#define LD(
 i) R##i=Simd::loadu(A+lda*i)
```

#### 13.178.2.3 ST

```
#define ST(
 i) Simd::storeu(B+ldb*i,R##i)
```

## 13.179 ffpack.dox File Reference

## 13.180 ffpack.h File Reference

Set of elimination based routines for dense linear algebra.

```
#include "givaro/givpoly1.h"
#include <fflas-ffpack/fflas-ffpack-config.h>
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/fflas/fflas_helpers.inl"
#include <list>
#include <vector>
#include <iostream>
#include <algorithm>
#include "fflas-ffpack/checkers/checkers_ffpack.h"
#include "ffpack_fgsv.inl"
#include "ffpack_fgetrs.inl"
#include "fflas-ffpack/checkers/checkers_ffpack.inl"
#include "ffpack_pluq.inl"
#include "ffpack_pluq_mp.inl"
#include "ffpack_ppluq.inl"
#include "ffpack_ludivine.inl"
#include "ffpack_ludivine_mp.inl"
#include "ffpack_echelonforms.inl"
#include "ffpack_fsytrf.inl"
#include "ffpack_invert.inl"
#include "ffpack_ftrtr.inl"
#include "ffpack_ftrstr.inl"
#include "ffpack_ftrssyr2k.inl"
#include "ffpack_charpoly_kglu.inl"
#include "ffpack_charpoly_kgfast.inl"
#include "ffpack_charpoly_kgfastgeneralized.inl"
#include "ffpack_charpoly_danilevski.inl"
#include "ffpack_charpoly.inl"
#include "ffpack_frobenius.inl"
```

```
#include "ffpack_minpoly.inl"
#include "ffpack_krylovelim.inl"
#include "ffpack_permutation.inl"
#include "ffpack_rankprofiles.inl"
#include "ffpack_det_mp.inl"
#include "ffpack_bruhatgen.inl"
#include "ffpack.inl"
```

## Data Structures

- class [CharpolyFailed](#)

## Namespaces

- namespace [FFPACK](#)  
*Finite Field PACK Set of elimination based routines for dense linear algebra.*
- namespace [FFPACK::Protected](#)

## Macros

- `#define __FFLASFFPACK_FTRSTR_THRESHOLD 64`
- `#define __FFLASFFPACK_FTRSSYR2K_THRESHOLD 64`

## Functions

- void [LAPACKPerm2MathPerm](#) (size\_t \*MathP, const size\_t \*LapackP, const size\_t N)  
*Conversion of a permutation from LAPACK format to Math format.*
- void [MathPerm2LAPACKPerm](#) (size\_t \*LapackP, const size\_t \*MathP, const size\_t N)  
*Conversion of a permutation from Maths format to LAPACK format.*
- template<class [Field](#) >  
void [applyP](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const [FFLAS::FFLAS\\_TRANSPOSE](#) Trans, const size\_t M, const size\_t ibeg, const size\_t iend, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t \*P)  
*Computes  $P1 \times \text{Diag}(L_R, P2)$  where  $P1$  is a LAPACK and  $P2$  a LAPACK permutation and store the result in  $P1$  as a LAPACK permutation.*
- template<class [Field](#) >  
void [applyP](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const [FFLAS::FFLAS\\_TRANSPOSE](#) Trans, const size\_t m, const size\_t ibeg, const size\_t iend, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t \*P, const [FFLAS::ParSeqHelper::Sequential](#) seq)
- template<class [Field](#) , class Cut , class Param >  
void [applyP](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const [FFLAS::FFLAS\\_TRANSPOSE](#) Trans, const size\_t m, const size\_t ibeg, const size\_t iend, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t \*P, const [FFLAS::ParSeqHelper::Parallel](#)< Cut, Param > par)
- template<class [Field](#) >  
void [MonotonicApplyP](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const [FFLAS::FFLAS\\_TRANSPOSE](#) Trans, const size\_t M, const size\_t ibeg, const size\_t iend, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t \*P, const size\_t R)  
*Apply a R-monotonically increasing permutation P, to the matrix A.*
- template<class [Field](#) >  
void [fgetrs](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, const size\_t R, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t \*P, const size\_t \*Q, typename [Field::Element\\_ptr](#) B, const size\_t ldb, int \*info)  
*Solve the system  $AX = B$  or  $XA = B$ .*



- `template<class Field >`  
`Field::Element_ptr fgetrs` (const `Field` &F, const `FFLAS::FFLAS_SIDE` Side, const `size_t` M, const `size_t` N, const `size_t` NRHS, const `size_t` R, typename `Field::Element_ptr` A, const `size_t` lda, const `size_t` \*P, const `size_t` \*Q, typename `Field::Element_ptr` X, const `size_t` ldx, typename `Field::ConstElement_ptr` B, const `size_t` ldb, int \*info)  
*Solve the system  $A X = B$  or  $X A = B$ .*
- `template<class Field >`  
`size_t fgesv` (const `Field` &F, const `FFLAS::FFLAS_SIDE` Side, const `size_t` M, const `size_t` N, typename `Field::Element_ptr` A, const `size_t` lda, typename `Field::Element_ptr` B, const `size_t` ldb, int \*info)  
*Square system solver.*
- `template<class Field >`  
`size_t fgesv` (const `Field` &F, const `FFLAS::FFLAS_SIDE` Side, const `size_t` M, const `size_t` N, const `size_t` NRHS, typename `Field::Element_ptr` A, const `size_t` lda, typename `Field::Element_ptr` X, const `size_t` ldx, typename `Field::ConstElement_ptr` B, const `size_t` ldb, int \*info)  
*Rectangular system solver.*
- `template<class Field >`  
`void ftrtri` (const `Field` &F, const `FFLAS::FFLAS_UPLO` Uplo, const `FFLAS::FFLAS_DIAG` Diag, const `size_t` N, typename `Field::Element_ptr` A, const `size_t` lda, const `size_t` threshold=\_\_FFLASFFPACK\_FTRTRI\_THRESHOLD)  
*Compute the inverse of a triangular matrix.*
- `template<class Field >`  
`void trinv_left` (const `Field` &F, const `size_t` N, typename `Field::ConstElement_ptr` L, const `size_t` ldl, typename `Field::Element_ptr` X, const `size_t` ldx)
- `template<class Field >`  
`void ftrtrm` (const `Field` &F, const `FFLAS::FFLAS_SIDE` side, const `FFLAS::FFLAS_DIAG` diag, const `size_t` N, typename `Field::Element_ptr` A, const `size_t` lda)  
*Compute the product of two triangular matrices of opposite shape.*
- `template<class Field >`  
`void ftrstr` (const `Field` &F, const `FFLAS::FFLAS_SIDE` side, const `FFLAS::FFLAS_UPLO` Uplo, const `FFLAS::FFLAS_DIAG` diagA, const `FFLAS::FFLAS_DIAG` diagB, const `size_t` N, typename `Field::ConstElement_ptr` A, const `size_t` lda, typename `Field::Element_ptr` B, const `size_t` ldb, const `size_t` threshold=\_\_FFLASFFPACK\_FTRSTR\_THRESHOLD)  
*Solve a triangular system with a triangular right hand side of the same shape.*
- `template<class Field >`  
`void ftrssyr2k` (const `Field` &F, const `FFLAS::FFLAS_UPLO` Uplo, const `FFLAS::FFLAS_DIAG` diagA, const `size_t` N, typename `Field::ConstElement_ptr` A, const `size_t` lda, typename `Field::Element_ptr` B, const `size_t` ldb, const `size_t` threshold=\_\_FFLASFFPACK\_FTRSSYR2K\_THRESHOLD)  
*Solve a triangular system in a symmetric sum: find B upper/lower triangular such that  $A^T B + B^T A = C$  where C is symmetric.*
- `template<class Field >`  
`bool fsytrf` (const `Field` &F, const `FFLAS::FFLAS_UPLO` UpLo, const `size_t` N, typename `Field::Element_ptr` A, const `size_t` lda, const `size_t` threshold=\_\_FFLASFFPACK\_FSYTRF\_THRESHOLD)  
*Triangular factorization of symmetric matrices.*
- `template<class Field >`  
`bool fsytrf` (const `Field` &F, const `FFLAS::FFLAS_UPLO` UpLo, const `size_t` N, typename `Field::Element_ptr` A, const `size_t` lda, const `FFLAS::ParSeqHelper::Sequential` seq, const `size_t` threshold=\_\_FFLASFFPACK\_FSYTRF\_THRESHOLD)
- `template<class Field, class Cut, class Param >`  
`bool fsytrf` (const `Field` &F, const `FFLAS::FFLAS_UPLO` UpLo, const `size_t` N, typename `Field::Element_ptr` A, const `size_t` lda, const `FFLAS::ParSeqHelper::Parallel`< Cut, Param > par, const `size_t` threshold=\_\_FFLASFFPACK\_FSYTRF\_THRESHOLD)
- `template<class Field >`  
`bool fsytrf_nonunit` (const `Field` &F, const `FFLAS::FFLAS_UPLO` UpLo, const `size_t` N, typename `Field::Element_ptr` A, const `size_t` lda, typename `Field::Element_ptr` D, const `size_t` incD, const `size_t` threshold=\_\_FFLASFFPACK\_FSYTRF\_THRESHOLD)  
*Triangular factorization of symmetric matrices.*
- `template<class Field >`  
`size_t PLUQ` (const `Field` &F, const `FFLAS::FFLAS_DIAG` Diag, const `size_t` M, const `size_t` N, typename `Field::Element_ptr` A, const `size_t` lda, `size_t` \*P, `size_t` \*Q)

*Compute a PLUQ factorization of the given matrix.*

- `template<class Field >`  
`size_t pPLUQ (const Field &F, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q)`
- `template<class Field >`  
`size_t PLUQ (const Field &F, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q, const FFLAS::ParSeqHelper::Sequential &PSHelper, size_t BCThreshold=__FFLASFFPACK_PLUQ_THRESHOLD)`
- `template<class Field , class Cut , class Param >`  
`size_t PLUQ (const Field &F, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q, const FFLAS::ParSeqHelper::Parallel< Cut, Param > &PSHelper)`
- `template<class Field >`  
`size_t LUdivine (const Field &F, const FFLAS::FFLAS_DIAG Diag, const FFLAS::FFLAS_TRANSPOSE trans, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const FFPACK_LU_TAG LuTag=FfpackSlabRecursive, const size_t cutoff=__FFLASFFPACK_LUDIVINE_THRESHOLD)`

*Compute the CUP or PLE factorization of the given matrix.*

- `template<class Field >`  
`size_t LUdivine_construct (const Field &F, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr X, const size_t ldx, typename Field::Element_ptr u, const size_t incu, size_t *P, bool computeX, const FFPACK_MINPOLY_TAG MinTag=FfpackDense, const size_t kg_mc=0, const size_t kg_mb=0, const size_t kg_j=0)`
- `template<class Field >`  
`size_t ColumnEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, bool transform=false, const FFPACK_LU_TAG LuTag=FfpackSlabRecursive)`

*Compute the Column Echelon form of the input matrix in-place.*

- `template<class Field >`  
`size_t pColumnEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, bool transform=false, size_t numthreads=0, const FFPACK_LU_TAG LuTag=FfpackTileRecursive)`
- `template<class Field , class PSHelper >`  
`size_t ColumnEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const bool transform, const FFPACK_LU_TAG LuTag, const PSHelper &psH)`
- `template<class Field >`  
`size_t RowEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const bool transform=false, const FFPACK_LU_TAG LuTag=FfpackSlabRecursive)`

*Compute the Row Echelon form of the input matrix in-place.*

- `template<class Field >`  
`size_t pRowEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const bool transform=false, size_t numthreads=0, const FFPACK_LU_TAG LuTag=FfpackTileRecursive)`
- `template<class Field , class PSHelper >`  
`size_t RowEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const bool transform, const FFPACK_LU_TAG LuTag, const PSHelper &psH)`
- `template<class Field >`  
`size_t ReducedColumnEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const bool transform=false, const FFPACK_LU_TAG LuTag=FfpackSlabRecursive)`

*Compute the Reduced Column Echelon form of the input matrix in-place.*

- `template<class Field >`  
`size_t pReducedColumnEchelonForm (const Field &F, const size_t M, const size_t N, typename`

- `Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const bool transform=false, size_t numthreads=0, const FFPACK_LU_TAG LuTag=FfpackTileRecursive)`
- `template<class Field, class PSHelper >`  
`size_t ReducedColumnEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const bool transform, const FFPACK_LU_TAG LuTag, const PSHelper &psH)`
  - `template<class Field >`  
`size_t ReducedRowEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const bool transform=false, const FFPACK_LU_TAG LuTag=FfpackSlabRecursive)`  
*Compute the Reduced Row Echelon form of the input matrix in-place.*
  - `template<class Field >`  
`size_t pReducedRowEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const bool transform=false, size_t numthreads=0, const FFPACK_LU_TAG LuTag=FfpackTileRecursive)`
  - `template<class Field, class PSHelper >`  
`size_t ReducedRowEchelonForm (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Qt, const bool transform, const FFPACK_LU_TAG LuTag, const PSHelper &psH)`
  - `template<class Field >`  
`size_t GaussJordan (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, const size_t colbeg, const size_t rowbeg, const size_t colsize, size_t *P, size_t *Q, const FFPACK_LU_TAG LuTag)`  
*Gauss-Jordan algorithm computing the Reduced Row echelon form and its transform matrix.*
  - `template<class Field >`  
`Field::Element_ptr Invert (const Field &F, const size_t M, typename Field::Element_ptr A, const size_t lda, int &>nullity)`  
*Invert the given matrix in place or computes its nullity if it is singular.*
  - `template<class Field >`  
`Field::Element_ptr Invert (const Field &F, const size_t M, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr X, const size_t ldx, int &>nullity)`  
*Invert the given matrix or computes its nullity if it is singular.*
  - `template<class Field >`  
`Field::Element_ptr Invert2 (const Field &F, const size_t M, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr X, const size_t ldx, int &>nullity)`  
*Invert the given matrix or computes its nullity if it is singular.*
  - `template<class PolRing >`  
`std::list< typename PolRing::Element > & CharPoly (const PolRing &R, std::list< typename PolRing::Element > &charp, const size_t N, typename PolRing::Domain_t::Element_ptr A, const size_t lda, typename PolRing::Domain_t::RandIter &G, const FFPACK_CHARPOLY_TAG CharpTag=FfpackAuto, const size_t degree=__FFLASFFPACK_ARITHPROG_THRESHOLD)`  
*Compute the characteristic polynomial of the matrix A.*
  - `template<class PolRing >`  
`PolRing::Element & CharPoly (const PolRing &R, typename PolRing::Element &charp, const size_t N, typename PolRing::Domain_t::Element_ptr A, const size_t lda, typename PolRing::Domain_t::RandIter &G, const FFPACK_CHARPOLY_TAG CharpTag=FfpackAuto, const size_t degree=__FFLASFFPACK_ARITHPROG_THRESHOLD)`  
*Compute the characteristic polynomial of the matrix A.*
  - `template<class PolRing >`  
`PolRing::Element & CharPoly (const PolRing &R, typename PolRing::Element &charp, const size_t N, typename PolRing::Domain_t::Element_ptr A, const size_t lda, const FFPACK_CHARPOLY_TAG CharpTag=FfpackAuto, const size_t degree=__FFLASFFPACK_ARITHPROG_THRESHOLD)`  
*Compute the characteristic polynomial of the matrix A.*
  - `template<class Field, class Polynomial >`  
`std::list< Polynomial > & KellerGehrig (const Field &F, std::list< Polynomial > &charp, const size_t N, typename Field::ConstElement_ptr A, const size_t lda)`

- `template<class Field , class Polynomial >`  
`int KGFast (const Field &F, std::list< Polynomial > &charp, const size_t N, typename Field::Element_ptr A,`  
`const size_t lda, size_t *kg_mc, size_t *kg_mb, size_t *kg_j)`
- `template<class Field , class Polynomial >`  
`std::list< Polynomial > & KGFast_generalized (const Field &F, std::list< Polynomial > &charp, const size_t`  
`N, typename Field::Element_ptr A, const size_t lda)`
- `template<class Field >`  
`void fgemv_kgf (const Field &F, const size_t N, typename Field::ConstElement_ptr A, const size_t lda, type-`  
`name Field::ConstElement_ptr X, const size_t incX, typename Field::Element_ptr Y, const size_t incY, const`  
`size_t kg_mc, const size_t kg_mb, const size_t kg_j)`
- `template<class Field , class Polynomial , class Randlter >`  
`std::list< Polynomial > & LUKrylov (const Field &F, std::list< Polynomial > &charp, const size_t N, typename`  
`Field::Element_ptr A, const size_t lda, typename Field::Element_ptr U, const size_t ldu, Randlter &G)`
- `template<class Field , class Polynomial >`  
`std::list< Polynomial > & Danilevski (const Field &F, std::list< Polynomial > &charp, const size_t N, type-`  
`name Field::Element_ptr A, const size_t lda)`
- `template<class PolRing >`  
`void RandomKrylovPrecond (const PolRing &PR, std::list< typename PolRing::Element > &completed,`  
`Factors, const size_t N, typename PolRing::Domain_t::Element_ptr A, const size_t lda, size_t &Nb, typename`  
`PolRing::Domain_t::Element_ptr &B, size_t &ldb, typename PolRing::Domain_t::Randlter &g, const size_t`  
`degree=__FFLASFFPACK_ARITHPROG_THRESHOLD)`
- `template<class PolRing >`  
`std::list< typename PolRing::Element > & ArithProg (const PolRing &PR, std::list< typename PolRing::`  
`Element > &frobeniusForm, const size_t N, typename PolRing::Domain_t::Element_ptr A, const size_t lda,`  
`const size_t degree)`
- `template<class Field , class Polynomial >`  
`std::list< Polynomial > & LUKrylov_KGFast (const Field &F, std::list< Polynomial > &charp, const size_t N,`  
`typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr X, const size_t ldx)`
- `template<class Field , class Polynomial >`  
`Polynomial & MinPoly (const Field &F, Polynomial &minP, const size_t N, typename Field::ConstElement_ptr`  
`A, const size_t lda)`  
*Compute the minimal polynomial of the matrix A.*
- `template<class Field , class Polynomial , class Randlter >`  
`Polynomial & MinPoly (const Field &F, Polynomial &minP, const size_t N, typename Field::ConstElement_ptr`  
`A, const size_t lda, Randlter &G)`  
*Compute the minimal polynomial of the matrix A.*
- `template<class Field , class Polynomial >`  
`Polynomial & MatVecMinPoly (const Field &F, Polynomial &minP, const size_t N, typename Field::ConstElement_ptr`  
`A, const size_t lda, typename Field::ConstElement_ptr v, const size_t incv)`  
*Compute the minimal polynomial of the matrix A and a vector v, namely the first linear dependency relation in the*  
*Krylov basis  $(v, Av, \dots, A^N v)$ .*
- `template<class Field , class Polynomial >`  
`Polynomial & MatVecMinPoly (const Field &F, Polynomial &minP, const size_t N, typename Field::ConstElement_ptr`  
`A, const size_t lda, typename Field::Element_ptr v, const size_t incv, typename Field::Element_ptr K, const`  
`size_t ldk, size_t *P)`
- `template<class Field , class Polynomial >`  
`Polynomial & Hybrid_KGF_LUK_MinPoly (const Field &F, Polynomial &minP, const size_t N, typename`  
`Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr X, const size_t ldx, size_t *P,`  
`const FFPACK_MINPOLY_TAG MinTag=FFPACK::FpackDense, const size_t kg_mc=0, const size_t kg_`  
`mb=0, const size_t kg_j=0)`
- `template<class Field >`  
`size_t Rank (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda)`  
*Computes the rank of the given matrix using a PLUQ factorization.*
- `template<class Field >`  
`size_t pRank (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda,`  
`size_t numthreads=0)`

- template<class [Field](#) , class PSHelper >  
 size\_t [Rank](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, const PSHelper &psH)  
*Returns true if the given matrix is singular.*
- template<class [Field](#) >  
 bool [IsSingular](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda)  
*Returns true if the given matrix is singular.*
- template<class [Field](#) >  
[Field::Element](#) & [Det](#) (const [Field](#) &F, typename [Field::Element](#) &det, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*P=NULL, size\_t \*Q=NULL)  
*Returns the determinant of the given square matrix.*
- template<class [Field](#) >  
[Field::Element](#) & [pDet](#) (const [Field](#) &F, typename [Field::Element](#) &det, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t numthreads=0, size\_t \*P=NULL, size\_t \*Q=NULL)
- template<class [Field](#) , class PSHelper >  
[Field::Element](#) & [Det](#) (const [Field](#) &F, typename [Field::Element](#) &det, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, const PSHelper &psH, size\_t \*P=NULL, size\_t \*Q=NULL)
- template<class [Field](#) >  
[Field::Element\\_ptr](#) [Solve](#) (const [Field](#) &F, const size\_t M, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) x, const int incx, typename [Field::ConstElement\\_ptr](#) b, const int incb)  
*Solves a linear system  $AX = b$  using PLUQ factorization.*
- template<class [Field](#) , class PSHelper >  
[Field::Element\\_ptr](#) [Solve](#) (const [Field](#) &F, const size\_t M, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) x, const int incx, typename [Field::ConstElement\\_ptr](#) b, const int incb, PSHelper &psH)
- template<class [Field](#) >  
[Field::Element\\_ptr](#) [pSolve](#) (const [Field](#) &F, const size\_t M, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) x, const int incx, typename [Field::ConstElement\\_ptr](#) b, const int incb, size\_t numthreads=0)
- template<class [Field](#) >  
 \*void [RandomNullSpaceVector](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) X, const size\_t incX)  
*Solve  $LX = B$  or  $XL = B$  in place.*
- template<class [Field](#) >  
 size\_t [NullSpaceBasis](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) &NS, size\_t &ldn, size\_t &NSdim)  
*Computes a basis of the Left/Right nullspace of the matrix A.*
- template<class [Field](#) >  
 size\_t [RowRankProfile](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*&rkprofile, const FFPACK\_LU\_TAG LuTag=[FfpackSlabRecursive](#))  
*Computes the row rank profile of A.*
- template<class [Field](#) >  
 size\_t [pRowRankProfile](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*&rkprofile, size\_t numthreads=0, const FFPACK\_LU\_TAG LuTag=[FfpackTileRecursive](#))
- template<class [Field](#) , class PSHelper >  
 size\_t [RowRankProfile](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*&rkprofile, const FFPACK\_LU\_TAG LuTag, PSHelper &psH)
- template<class [Field](#) >  
 size\_t [ColumnRankProfile](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*&rkprofile, const FFPACK\_LU\_TAG LuTag=[FfpackSlabRecursive](#))  
*Computes the column rank profile of A.*
- template<class [Field](#) >  
 size\_t [pColumnRankProfile](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*&rkprofile, size\_t numthreads=0, const FFPACK\_LU\_TAG LuTag=[FfpackTileRecursive](#))

- `template<class Field, class PSHelper >`  
`size_t ColumnRankProfile (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A,`  
`const size_t lda, size_t *rkprofile, const FFPACK_LU_TAG LuTag, PSHelper &psH)`
- `void RankProfileFromLU (const size_t *P, const size_t N, const size_t R, size_t *rkprofile, const FFPACK_↵`  
`LU_TAG LuTag)`  
*Recovers the column/row rank profile from the permutation of an LU decomposition.*
- `size_t LeadingSubmatrixRankProfiles (const size_t M, const size_t N, const size_t R, const size_t LSm, const`  
`size_t LSn, const size_t *P, const size_t *Q, size_t *RRP, size_t *CRP)`  
*Recovers the row and column rank profiles of any leading submatrix from the PLUQ decomposition.*
- `template<class Field >`  
`size_t RowRankProfileSubmatrixIndices (const Field &F, const size_t M, const size_t N, typename`  
`Field::Element_ptr A, const size_t lda, size_t *rowindices, size_t *colindices, size_t &R)`  
*RowRankProfileSubmatrixIndices.*
- `template<class Field >`  
`size_t ColRankProfileSubmatrixIndices (const Field &F, const size_t M, const size_t N, typename`  
`Field::Element_ptr A, const size_t lda, size_t *rowindices, size_t *colindices, size_t &R)`  
*Computes the indices of the submatrix  $r \times r$   $X$  of  $A$  whose columns correspond to the column rank profile of  $A$ .*
- `template<class Field >`  
`size_t RowRankProfileSubmatrix (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr`  
`A, const size_t lda, typename Field::Element_ptr &X, size_t &R)`  
*Computes the  $r \times r$  submatrix  $X$  of  $A$ , by picking the row rank profile rows of  $A$ .*
- `template<class Field >`  
`size_t ColRankProfileSubmatrix (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr`  
`A, const size_t lda, typename Field::Element_ptr &X, size_t &R)`  
*Compute the  $r \times r$  submatrix  $X$  of  $A$ , by picking the row rank profile rows of  $A$ .*
- `template<class Field >`  
`void getTriangular (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG diag, const`  
`size_t M, const size_t N, const size_t R, typename Field::ConstElement_ptr A, const size_t lda, typename`  
`Field::Element_ptr T, const size_t ldt, const bool OnlyNonZeroVectors=false)`  
*Extracts a triangular matrix from a compact storage  $A=L\backslash U$  of rank  $R$ .*
- `template<class Field >`  
`void getTriangular (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG diag, const`  
`size_t M, const size_t N, const size_t R, typename Field::Element_ptr A, const size_t lda)`  
*Cleans up a compact storage  $A=L\backslash U$  to reveal a triangular matrix of rank  $R$ .*
- `template<class Field >`  
`void getEchelonForm (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG diag,`  
`const size_t M, const size_t N, const size_t R, const size_t *P, typename Field::ConstElement_ptr A, const`  
`size_t lda, typename Field::Element_ptr T, const size_t ldt, const bool OnlyNonZeroVectors=false, const`  
`FFPACK_LU_TAG LuTag=FpackSlabRecursive)`  
*Extracts a matrix in echelon form from a compact storage  $A=L\backslash U$  of rank  $R$  obtained by RowEchelonForm or Column↵*  
*EchelonForm.*
- `template<class Field >`  
`void getEchelonForm (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG diag,`  
`const size_t M, const size_t N, const size_t R, const size_t *P, typename Field::Element_ptr A, const size_t`  
`lda, const FFPACK_LU_TAG LuTag=FpackSlabRecursive)`  
*Cleans up a compact storage  $A=L\backslash U$  obtained by RowEchelonForm or ColumnEchelonForm to reveal an echelon*  
*form of rank  $R$ .*
- `template<class Field >`  
`void getEchelonTransform (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG`  
`diag, const size_t M, const size_t N, const size_t R, const size_t *P, const size_t *Q, typename`  
`Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr T, const size_t ldt, const FFPACK_↵`  
`LU_TAG LuTag=FpackSlabRecursive)`  
*Extracts a transformation matrix to echelon form from a compact storage  $A=L\backslash U$  of rank  $R$  obtained by RowEchelon↵*  
*Form or ColumnEchelonForm.*



- `template<class Field >`  
`void getReducedEchelonForm (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const size_t M, const size_t N, const size_t R, const size_t *P, typename Field::ConstElement_ptr A, const size_t Ida, typename Field::Element_ptr T, const size_t ldt, const bool OnlyNonZeroVectors=false, const FFPACK_LU_TAG LuTag=FFpackSlabRecursive)`  
*Extracts a matrix in echelon form from a compact storage  $A=L\backslash U$  of rank  $R$  obtained by ReducedRowEchelonForm or ReducedColumnEchelonForm with transform = true.*
- `template<class Field >`  
`void getReducedEchelonForm (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const size_t M, const size_t N, const size_t R, const size_t *P, typename Field::Element_ptr A, const size_t Ida, const FFPACK_LU_TAG LuTag=FFpackSlabRecursive)`  
*Cleans up a compact storage  $A=L\backslash U$  of rank  $R$  obtained by ReducedRowEchelonForm or ReducedColumnEchelonForm with transform = true.*
- `template<class Field >`  
`void getReducedEchelonTransform (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const size_t M, const size_t N, const size_t R, const size_t *P, const size_t *Q, typename Field::ConstElement_ptr A, const size_t Ida, typename Field::Element_ptr T, const size_t ldt, const FFPACK_LU_TAG LuTag=FFpackSlabRecursive)`  
*Extracts a transformation matrix to echelon form from a compact storage  $A=L\backslash U$  of rank  $R$  obtained by RowEchelonForm or ColumnEchelonForm.*
- `void PLUQtoEchelonPermutation (const size_t N, const size_t R, const size_t *P, size_t *outPerm)`  
*Auxiliary routine: determines the permutation that changes a PLUQ decomposition into a echelon form revealing PLUQ decomposition.*
- `template<class Field >`  
`size_t LTBruhatGen (const Field &Fi, const FFLAS::FFLAS_DIAG diag, const size_t N, typename Field::Element_ptr A, const size_t Ida, size_t *P, size_t *Q)`  
*LTBruhatGen Suppose A is Left Triangular Matrix This procedure computes the Bruhat Representation of A and return the rank of A.*
- `template<class Field >`  
`void getLTBruhatGen (const Field &Fi, const size_t N, const size_t r, const size_t *P, const size_t *Q, typename Field::Element_ptr R, const size_t ldr)`  
*GetLTBruhatGen This procedure Computes the Rank Revealing Matrix based on the Bruhta representation of a Matrix.*
- `template<class Field >`  
`void getLTBruhatGen (const Field &Fi, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG diag, const size_t N, const size_t r, const size_t *P, const size_t *Q, typename Field::ConstElement_ptr A, const size_t Ida, typename Field::Element_ptr T, const size_t ldt)`  
*GetLTBruhatGen This procedure computes the matrix L or U f the Bruhat Representation Suppose that A is the bruhat representation of a matrix.*
- `size_t LTQSorder (const size_t N, const size_t r, const size_t *P, const size_t *Q)`  
*LTQSorder This procedure computes the order of quasiseparability of a matrix.*
- `template<class Field >`  
`size_t CompressToBlockBiDiagonal (const Field &Fi, const FFLAS::FFLAS_UPLO Uplo, size_t N, size_t s, size_t r, const size_t *P, const size_t *Q, typename Field::Element_ptr A, size_t Ida, typename Field::Element_ptr X, size_t ldx, size_t *K, size_t *M, size_t *T)`  
*CompressToBlockBiDiagonal This procedure compress a compact representation of a row echelon form or column echelon form.*
- `template<class Field >`  
`void ExpandBlockBiDiagonalToBruhat (const Field &Fi, const FFLAS::FFLAS_UPLO Uplo, size_t N, size_t s, size_t r, typename Field::Element_ptr A, size_t Ida, typename Field::Element_ptr X, size_t ldx, size_t NbBlocks, size_t *K, size_t *M, size_t *T)`  
*ExpandBlockBiDiagonal This procedure expand a compact representation of a row echelon form or column echelon form.*
- `void Bruhat2EchelonPermutation (size_t N, size_t R, const size_t *P, const size_t *Q, size_t *M)`  
*Bruhat2EchelonPermutation (N,R,P,Q) Compute M such that LM or MU is in echelon form where L or U are factors of the Bruhat Rpresentation.*
- `size_t * Tinverter (size_t *T, size_t r)`

- `template<class Field >`  
`void ComputeRPermutation (const Field &Fi, size_t N, size_t r, const size_t *P, const size_t *Q, size_t *R, size_t *MU, size_t *ML)`
- `template<class Field >`  
`void productBruhatxTS (const Field &Fi, size_t N, size_t s, size_t r, const size_t *P, const size_t *Q, const typename Field::Element_ptr Xu, size_t ldu, size_t NbBlocksU, size_t *Ku, size_t *Tu, size_t *MU, const typename Field::Element_ptr XI, size_t ldl, size_t NbBlocksL, size_t *KI, size_t *TI, size_t *ML, typename Field::Element_ptr B, size_t t, size_t ldb, typename Field::Element_ptr C, size_t ldc)`  
*productBruhatxTS Compute the product between the CRE compact representation of a matrix A and B a tall matrix*
- `template<class Field >`  
`Field::Element_ptr LQUPtoInverseOfFullRankMinor (const Field &F, const size_t rank, typename Field::Element_ptr A_factors, const size_t lda, const size_t *QtPointer, typename Field::Element_ptr X, const size_t ldx)`  
*LQUPtoInverseOfFullRankMinor.*

### 13.180.1 Detailed Description

Set of elimination based routines for dense linear algebra.

Matrices are supposed over finite prime field of characteristic less than  $2^{26}$ .

### 13.180.2 Macro Definition Documentation

#### 13.180.2.1 \_\_FFLASFFPACK\_FTRSTR\_THRESHOLD

```
#define __FFLASFFPACK_FTRSTR_THRESHOLD 64
```

#### 13.180.2.2 \_\_FFLASFFPACK\_FTRSSYR2K\_THRESHOLD

```
#define __FFLASFFPACK_FTRSSYR2K_THRESHOLD 64
```

## 13.181 ffpack.inl File Reference

### Namespaces

- namespace `FFPACK`  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

### Macros

- `#define __FFLASFFPACK_ffpack_INL`

### Functions

- `template<class Field >`  
`size_t Rank (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda)`  
*Computes the rank of the given matrix using a PLUQ factorization.*
- `template<class Field >`  
`size_t pRank (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t numthreads=0)`
- `template<class Field , class PSHelper >`  
`size_t Rank (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, const PSHelper &psH)`
- `template<class Field >`  
`bool IsSingular (const Field &F, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda)`  
*Returns true if the given matrix is singular.*



- template<class [Field](#) >  
[Field::Element](#) & [Det](#) (const [Field](#) &F, typename [Field::Element](#) &det, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*P=NULL, size\_t \*Q=NULL)  
*Returns the determinant of the given square matrix.*
- template<class [Field](#) >  
[Field::Element](#) & [pDet](#) (const [Field](#) &F, typename [Field::Element](#) &det, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t numthreads=0, size\_t \*P=NULL, size\_t \*Q=NULL)
- template<class [Field](#) , class PSHelper >  
[Field::Element](#) & [Det](#) (const [Field](#) &F, typename [Field::Element](#) &det, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, const PSHelper &psH, size\_t \*P=NULL, size\_t \*Q=NULL)
- template<class [Field](#) >  
[Field::Element\\_ptr](#) [Solve](#) (const [Field](#) &F, const size\_t M, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) x, const int incx, typename [Field::ConstElement\\_ptr](#) b, const int incb)  
*Solves a linear system  $AX = b$  using PLUQ factorization.*
- template<class [Field](#) , class PSHelper >  
[Field::Element\\_ptr](#) [Solve](#) (const [Field](#) &F, const size\_t M, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) x, const int incx, typename [Field::ConstElement\\_ptr](#) b, const int incb, PSHelper &psH)
- template<class [Field](#) >  
[Field::Element\\_ptr](#) [pSolve](#) (const [Field](#) &F, const size\_t M, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) x, const int incx, typename [Field::ConstElement\\_ptr](#) b, const int incb, size\_t numthreads=0)
- template<class [Field](#) >  
void [RandomNullSpaceVector](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) X, const size\_t incX)  
*Solve  $LX = B$  or  $XL = B$  in place.*
- template<class [Field](#) >  
size\_t [NullSpaceBasis](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) &NS, size\_t &ldn, size\_t &NSdim)  
*Computes a basis of the Left/Right nullspace of the matrix A.*
- template<class [Field](#) >  
void [solveLB](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, const size\_t R, typename [Field::Element\\_ptr](#) L, const size\_t ldl, const size\_t \*Q, typename [Field::Element\\_ptr](#) B, const size\_t ldb)
- template<class [Field](#) >  
void [solveLB2](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, const size\_t R, typename [Field::Element\\_ptr](#) L, const size\_t ldl, const size\_t \*Q, typename [Field::Element\\_ptr](#) B, const size\_t ldb)

## 13.181.1 Macro Definition Documentation

### 13.181.1.1 \_\_FFLASFFPACK\_ffpack\_INL

```
#define __FFLASFFPACK_ffpack_INL
```

## 13.182 ffpack\_bruhatgen.inl File Reference

### Namespaces

- namespace [FFPACK](#)  
*Finite **Field** **PACK** Set of elimination based routines for dense linear algebra.*

### Macros

- #define [\\_\\_FFLASFFPACK\\_ffpack\\_bruhatgen\\_inl](#)

## Functions

- `template<class Field >`  
`size_t LTBruhatGen (const Field &Fi, const FFLAS::FFLAS_DIAG diag, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q)`  
*LTBruhatGen Suppose A is Left Triangular Matrix This procedure computes the Bruhat Representation of A and return the rank of A.*
- `template<class Field >`  
`void getLTBruhatGen (const Field &Fi, const size_t N, const size_t r, const size_t *P, const size_t *Q, typename Field::Element_ptr R, const size_t ldr)`  
*GetLTBruhatGen This procedure Computes the Rank Revealing Matrix based on the Bruhta representation of a Matrix.*
- `template<class Field >`  
`void getLTBruhatGen (const Field &Fi, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG diag, const size_t N, const size_t r, const size_t *P, const size_t *Q, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr T, const size_t ldt)`  
*GetLTBruhatGen This procedure computes the matrix L or U f the Bruhat Representation Suppose that A is the bruhat representation of a matrix.*
- `size_t LTQSorder (const size_t N, const size_t r, const size_t *P, const size_t *Q)`  
*LTQSorder This procedure computes the order of quasiseparability of a matrix.*
- `template<class Field >`  
`size_t CompressToBlockBiDiagonal (const Field &Fi, const FFLAS::FFLAS_UPLO Uplo, size_t N, size_t s, size_t r, const size_t *P, const size_t *Q, typename Field::Element_ptr A, size_t lda, typename Field::Element_ptr X, size_t ldx, size_t *K, size_t *M, size_t *T)`  
*CompressToBlockBiDiagonal This procedure compress a compact representation of a row echelon form or column echelon form.*
- `template<class Field >`  
`void ExpandBlockBiDiagonalToBruhat (const Field &Fi, const FFLAS::FFLAS_UPLO Uplo, size_t N, size_t s, size_t r, typename Field::Element_ptr A, size_t lda, typename Field::Element_ptr X, size_t ldx, size_t NbBlocks, size_t *K, size_t *M, size_t *T)`  
*ExpandBlockBiDiagonal This procedure expand a compact representation of a row echelon form or column echelon form.*
- `void Bruhat2EchelonPermutation (size_t N, size_t R, const size_t *P, const size_t *Q, size_t *M)`  
*Bruhat2EchelonPermutation (N,R,P,Q) Compute M such that LM or MU is in echelon form where L or U are factors of the Bruhat Rpresentation.*
- `size_t * Tinverter (const size_t *T, size_t r)`
- `template<class Field >`  
`void ComputeRPermutation (const Field &Fi, size_t N, size_t r, const size_t *P, const size_t *Q, size_t *R, const size_t *MU, const size_t *ML)`
- `template<class Field >`  
`Field::Element_ptr expandLCRE (const Field &Fi, size_t N, size_t s, size_t r, size_t *R, size_t i, typename Field::ConstElement_ptr Xu, size_t ldu, size_t NbBlocksU, const size_t *Ku, const size_t *Tuinv, typename Field::ConstElement_ptr Xl, size_t ldl, size_t NbBlocksL, const size_t *Kl, const size_t *Tlinv, typename Field::Element_ptr CRE, size_t ldcre)`  
*Expands an anti-diagonal block of a left triangular matrix from its compact Bruhat representation.*
- `template<class Field >`  
`void productBruhatxTS (const Field &Fi, size_t N, size_t s, size_t r, size_t t, const size_t *P, const size_t *Q, typename Field::ConstElement_ptr Xu, size_t ldu, size_t NbBlocksU, const size_t *Ku, const size_t *Tu, const size_t *MU, typename Field::ConstElement_ptr Xl, size_t ldl, size_t NbBlocksL, const size_t *Kl, const size_t *Tl, const size_t *ML, typename Field::Element_ptr B, size_t ldb, const typename Field::Element beta, typename Field::Element_ptr D, size_t ldd)`  
*Compute the product of a left-triangular quasi-separable matrix A, represented by a compact Bruhat generator, with a dense rectangular matrix B:  $C \leftarrow A \times B + \text{beta}C$ .*

## 13.182.1 Macro Definition Documentation

### 13.182.1.1 \_\_FFLASFFPACK\_ffpack\_bruhatgen\_inl

```
#define __FFLASFFPACK_ffpack_bruhatgen_inl
```

## 13.183 ffpack\_charpoly.inl File Reference

```
#include "fflas-ffpack/utils/fflas_randommatrix.h"
#include "ffpack_charpoly_mp.inl"
```

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*
- namespace [FFPACK::Protected](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_charpoly\\_INL](#)

### Functions

- template<class PolRing >  
std::list< typename PolRing::Element > & [CharPoly](#) (const PolRing &R, std::list< typename PolRing::Element > &charp, const size\_t N, typename PolRing::Domain\_t::Element\_ptr A, const size\_t lda, typename PolRing::Domain\_t::RandIter &G, const FFPACK\_CHARPOLY\_TAG CharpTag=FfpackAuto, const size\_t degree=\_\_FFLASFFPACK\_ARITHPROG\_THRESHOLD)  
*Compute the characteristic polynomial of the matrix A.*
- template<class PolRing >  
PolRing::Element & [CharPoly](#) (const PolRing &R, typename PolRing::Element &charp, const size\_t N, typename PolRing::Domain\_t::Element\_ptr A, const size\_t lda, typename PolRing::Domain\_t::RandIter &G, const FFPACK\_CHARPOLY\_TAG CharpTag=FfpackAuto, const size\_t degree=\_\_FFLASFFPACK\_ARITHPROG\_THRESHOLD)  
*Compute the characteristic polynomial of the matrix A.*
- template<class [Field](#) , class Polynomial , class RandIter >  
std::list< Polynomial > & [LUKrylov](#) (const [Field](#) &F, std::list< Polynomial > &charp, const size\_t N, typename [Field](#)::Element\_ptr A, const size\_t lda, typename [Field](#)::Element\_ptr U, const size\_t ldu, RandIter &G)
- template<class [Field](#) , class Polynomial >  
std::list< Polynomial > & [LUKrylov\\_KGFast](#) (const [Field](#) &F, std::list< Polynomial > &charp, const size\_t N, typename [Field](#)::Element\_ptr A, const size\_t lda, typename [Field](#)::Element\_ptr X, const size\_t ldx)

## 13.183.1 Macro Definition Documentation

### 13.183.1.1 \_\_FFLASFFPACK\_charpoly\_INL

```
#define __FFLASFFPACK_charpoly_INL
```

## 13.184 ffpack\_charpoly\_danilevski.inl File Reference

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

### Macros

- #define [\\_\\_FFLASFFPACK\\_ffpack\\_charpoly\\_danilveski\\_INL](#)

## Functions

- `template<class Field , class Polynomial >`  
`std::list< Polynomial > & Danilevski (const Field &F, std::list< Polynomial > &charp, const size_t N, type-`  
`name Field::Element\_ptr A, const size_t lda)`

## 13.184.1 Macro Definition Documentation

### 13.184.1.1 `__FFLASFFPACK_ffpack_charpoly_danilveski_INL`

```
#define __FFLASFFPACK_ffpack_charpoly_danilveski_INL
```

## 13.185 `ffpack_charpoly_kgfast.inl` File Reference

### Namespaces

- namespace [FFPACK](#)  
*Finite **Field** **PACK** Set of elimination based routines for dense linear algebra.*
- namespace [FFPACK::Protected](#)

### Macros

- `#define \_\_FFLASFFPACK\_ffpack\_charpoly\_kgfast\_INL`

### Functions

- `template<class Field , class Polynomial >`  
`int KGFast (const Field &F, std::list< Polynomial > &charp, const size_t N, typename Field::Element\_ptr A,`  
`const size_t lda, size_t *kg_mc, size_t *kg_mb, size_t *kg_j)`
- `template<class Field >`  
`void fgemv\_kgf (const Field &F, const size_t N, typename Field::ConstElement\_ptr A, const size_t lda, type-`  
`name Field::ConstElement\_ptr X, const size_t incX, typename Field::Element\_ptr Y, const size_t incY, const`  
`size_t kg_mc, const size_t kg_mb, const size_t kg_j)`

## 13.185.1 Macro Definition Documentation

### 13.185.1.1 `__FFLASFFPACK_ffpack_charpoly_kgfast_INL`

```
#define __FFLASFFPACK_ffpack_charpoly_kgfast_INL
```

## 13.186 `ffpack_charpoly_kgfastgeneralized.inl` File Reference

```
#include <iostream>
#include "fflas-ffpack/utils/fflas_io.h"
```

### Namespaces

- namespace [FFPACK](#)  
*Finite **Field** **PACK** Set of elimination based routines for dense linear algebra.*
- namespace [FFPACK::Protected](#)

### Macros

- `#define \_\_FFLASFFPACK\_ffpack\_charpoly\_kgfastgeneralized\_INL`

## Functions

- template<class [Field](#) >  
[Field::Element\\_ptr](#) buildMatrix (const [Field](#) &F, typename [Field::ConstElement\\_ptr](#) E, typename [Field::ConstElement\\_ptr](#) C, const size\_t lda, const size\_t \*B, const size\_t \*T, const size\_t me, const size\_t mc, const size\_t lambda, const size\_t mu)
- template<class [Field](#) , class Polynomial >  
std::list< Polynomial > & [KGFast\\_generalized](#) (const [Field](#) &F, std::list< Polynomial > &charp, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda)

### 13.186.1 Macro Definition Documentation

#### 13.186.1.1 \_\_FFLASFFPACK\_ffpack\_charpoly\_kgfastgeneralized\_INL

```
#define __FFLASFFPACK_ffpack_charpoly_kgfastgeneralized_INL
```

## 13.187 ffpack\_charpoly\_kglu.inl File Reference

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*
- namespace [FFPACK::Protected](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_ffpack\\_charpoly\\_kglu\\_INL](#)

## Functions

- template<class [Field](#) >  
size\_t [updated](#) (const [Field](#) &F, size\_t \*d, size\_t k, std::vector< std::vector< typename [Field::Element](#) > > &minpt)
- template<class [Field](#) >  
size\_t [newD](#) (const [Field](#) &F, size\_t \*d, bool &KeepOn, const size\_t l, const size\_t N, typename [Field::Element\\_ptr](#) X, const size\_t \*Q, std::vector< std::vector< typename [Field::Element](#) > > &minpt)
- template<class [Field](#) , class Polynomial >  
std::list< Polynomial > & [KellerGehrig](#) (const [Field](#) &F, std::list< Polynomial > &charp, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda)

### 13.187.1 Macro Definition Documentation

#### 13.187.1.1 \_\_FFLASFFPACK\_ffpack\_charpoly\_kglu\_INL

```
#define __FFLASFFPACK_ffpack_charpoly_kglu_INL
```

## 13.188 ffpack\_charpoly\_mp.inl File Reference

```
#include <givaro/zring.h>
#include "givaro/givinteger.h"
#include "givaro/givpoly1.h"
#include "fflas-ffpack/field/rns-integer.h"
#include "fflas-ffpack/fflas-ffpack.h"
```

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

## Macros

- `#define __FFPACK_charpoly_mp_INL`

## Functions

- `FFPACK::RNSInteger< FFPACK::rns_double >::Element_ptr CharPoly` (const `FFPACK::RNSInteger< FFPACK::rns_double > &F`, typename `FFPACK::RNSInteger< FFPACK::rns_double >::Element_ptr` charp, const `size_t N`, typename `FFPACK::RNSInteger< FFPACK::rns_double >::Element_ptr` A, const `size_t lda`, `Givaro::ZRing< Givaro::Integer >::RandIter &G`, const `FFPACK_CHARPOLY_TAG` CharpTag, `size_t degree`)
- `template<> Givaro::Poly1Dom< Givaro::ZRing< Givaro::Integer > >::Element & CharPoly` (const `Givaro::Poly1Dom< Givaro::ZRing< Givaro::Integer > > &R`, `Givaro::Poly1Dom< Givaro::ZRing< Givaro::Integer > >::Element &charp`, const `size_t N`, `Givaro::Integer *A`, const `size_t lda`, `Givaro::ZRing< Givaro::Integer >::RandIter &G`, const `FFPACK_CHARPOLY_TAG` CharpTag, `size_t degree`)

## 13.188.1 Macro Definition Documentation

### 13.188.1.1 \_\_FFPACK\_charpoly\_mp\_INL

```
#define __FFPACK_charpoly_mp_INL
```

## 13.189 ffpack\_det\_mp.inl File Reference

```
#include <givaro/zring.h>
#include "givaro/givinteger.h"
#include "fflas-ffpack/field/rns-integer.h"
#include "fflas-ffpack/fflas-ffpack.h"
```

## Namespaces

- namespace `FFPACK`  
*Finite Field PACK Set of elimination based routines for dense linear algebra.*

## Macros

- `#define __FFPACK_det_mp_INL`

## Functions

- `template<class PSHelper > FFPACK::RNSInteger< FFPACK::rns_double >::Element_ptr & Det` (const `FFPACK::RNSInteger< FFPACK::rns_double > &F`, typename `FFPACK::RNSInteger< FFPACK::rns_double >::Element_ptr` &det, const `size_t N`, typename `FFPACK::RNSInteger< FFPACK::rns_double >::Element_ptr` A, const `size_t lda`, const `PSHelper &psH`)
- `template<class PSHelper > Givaro::Integer & Det` (const `Givaro::ZRing< Givaro::Integer > &F`, `Givaro::Integer &det`, const `size_t N`, `Givaro::Integer *A`, const `size_t lda`, const `PSHelper &psH`, `size_t *P`, `size_t *Q`)

## 13.189.1 Macro Definition Documentation

### 13.189.1.1 \_\_FFPACK\_det\_mp\_INL

```
#define __FFPACK_det_mp_INL
```

## 13.190 ffpack\_echelonforms.inl File Reference

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

### Macros

- #define [\\_\\_FFLASFFPACK\\_ffpack\\_echelon\\_forms\\_INL](#)
- #define [\\_\\_FFLASFFPACK\\_GAUSSJORDAN\\_BASECASE](#) 256

### Functions

- template<class [Field](#) >  
void [getTriangular](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_UPLO](#) Uplo, const [FFLAS::FFLAS\\_DIAG](#) diag, const size\_t M, const size\_t N, const size\_t R, typename [Field::ConstElement\\_ptr](#) A, const size\_t Ida, typename [Field::Element\\_ptr](#) T, const size\_t Idt, const bool OnlyNonZeroVectors=false)  
*Extracts a triangular matrix from a compact storage  $A=L\backslash U$  of rank  $R$ .*
- template<class [Field](#) >  
void [getTriangular](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_UPLO](#) Uplo, const [FFLAS::FFLAS\\_DIAG](#) diag, const size\_t M, const size\_t N, const size\_t R, typename [Field::Element\\_ptr](#) A, const size\_t Ida)  
*Cleans up a compact storage  $A=L\backslash U$  to reveal a triangular matrix of rank  $R$ .*
- void [PLUQtoEchelonPermutation](#) (const size\_t N, const size\_t R, const size\_t \*P, size\_t \*outPerm)  
*Auxiliary routine: determines the permutation that changes a PLUQ decomposition into a echelon form revealing PLUQ decomposition.*
- template<class [Field](#) >  
void [getEchelonForm](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_UPLO](#) Uplo, const [FFLAS::FFLAS\\_DIAG](#) diag, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, typename [Field::ConstElement\\_ptr](#) A, const size\_t Ida, typename [Field::Element\\_ptr](#) T, const size\_t Idt, const bool OnlyNonZeroVectors=false, const [FFPACK\\_LU\\_TAG](#) LuTag=[FfpackSlabRecursive](#))  
*Extracts a matrix in echelon form from a compact storage  $A=L\backslash U$  of rank  $R$  obtained by RowEchelonForm or Column↔EchelonForm.*
- template<class [Field](#) >  
void [getEchelonForm](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_UPLO](#) Uplo, const [FFLAS::FFLAS\\_DIAG](#) diag, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, typename [Field::Element\\_ptr](#) A, const size\_t Ida, const [FFPACK\\_LU\\_TAG](#) LuTag=[FfpackSlabRecursive](#))  
*Cleans up a compact storage  $A=L\backslash U$  obtained by RowEchelonForm or ColumnEchelonForm to reveal an echelon form of rank  $R$ .*
- template<class [Field](#) >  
void [getEchelonTransform](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_UPLO](#) Uplo, const [FFLAS::FFLAS\\_DIAG](#) diag, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, const size\_t \*Q, typename [Field::ConstElement\\_ptr](#) A, const size\_t Ida, typename [Field::Element\\_ptr](#) T, const size\_t Idt, const [FFPACK\\_LU\\_TAG](#) LuTag=[FfpackSlabRecursive](#))  
*Extracts a transformation matrix to echelon form from a compact storage  $A=L\backslash U$  of rank  $R$  obtained by RowEchelon↔Form or ColumnEchelonForm.*
- template<class [Field](#) >  
void [getReducedEchelonForm](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_UPLO](#) Uplo, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, typename [Field::ConstElement\\_ptr](#) A, const size\_t Ida, typename [Field::Element\\_ptr](#) T, const size\_t Idt, const bool OnlyNonZeroVectors=false, const [FFPACK\\_LU\\_TAG](#) LuTag=[FfpackSlabRecursive](#))  
*Extracts a matrix in echelon form from a compact storage  $A=L\backslash U$  of rank  $R$  obtained by ReducedRowEchelonForm or ReducedColumnEchelonForm with transform = true.*
- template<class [Field](#) >  
void [getReducedEchelonForm](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_UPLO](#) Uplo, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, typename [Field::Element\\_ptr](#) A, const size\_t Ida, const [FFPACK\\_LU\\_TAG](#) LuTag=[FfpackSlabRecursive](#))

*Cleans up a compact storage  $A=L\backslash U$  of rank  $R$  obtained by `ReducedRowEchelonForm` or `ReducedColumnEchelonForm` with `transform = true`.*

- `template<class Field >`  
`void getReducedEchelonTransform (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const size_t M, const size_t N, const size_t R, const size_t *P, const size_t *Q, typename Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr T, const size_t ldt, const FFPACK_LU_TAG LuTag=FpackSlabRecursive)`

*Extracts a transformation matrix to echelon form from a compact storage  $A=L\backslash U$  of rank  $R$  obtained by `RowEchelonForm` or `ColumnEchelonForm`.*

## 13.190.1 Macro Definition Documentation

### 13.190.1.1 \_\_FFLASFFPACK\_ffpack\_echelon\_forms\_INL

```
#define __FFLASFFPACK_ffpack_echelon_forms_INL
```

### 13.190.1.2 \_\_FFLASFFPACK\_GAUSSJORDAN\_BASECASE

```
#define __FFLASFFPACK_GAUSSJORDAN_BASECASE 256
```

## 13.191 fpack\_fgesv.inl File Reference

### Namespaces

- namespace `FFPACK`  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

### Macros

- `#define __FFLASFFPACK_ffpack_fgesv_INL`

### Functions

- `template<class Field >`  
`size_t fgesv (const Field &F, const FFLAS::FFLAS_SIDE Side, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb, int *info)`  
*Square system solver.*
- `template<class Field >`  
`size_t fgesv (const Field &F, const FFLAS::FFLAS_SIDE Side, const size_t M, const size_t N, const size_t NRHS, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr X, const size_t ldx, typename Field::ConstElement_ptr B, const size_t ldb, int *info)`  
*Rectangular system solver.*

## 13.191.1 Macro Definition Documentation

### 13.191.1.1 \_\_FFLASFFPACK\_ffpack\_fgesv\_INL

```
#define __FFLASFFPACK_ffpack_fgesv_INL
```

## 13.192 fpack\_fgetrs.inl File Reference

### Namespaces

- namespace `FFPACK`  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

### Macros

- `#define __FFLASFFPACK_ffpack_fgetrs_INL`



## Functions

- template<class [Field](#) >  
void [fgetrs](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, const size\_t R, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t \*P, const size\_t \*Q, typename [Field::Element\\_ptr](#) B, const size\_t ldb, int \*info)  
*Solve the system  $AX = B$  or  $XA = B$ .*
- template<class [Field](#) >  
[Field::Element\\_ptr](#) [fgetrs](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, const size\_t NRHS, const size\_t R, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t \*P, const size\_t \*Q, typename [Field::Element\\_ptr](#) X, const size\_t idx, typename [Field::ConstElement\\_ptr](#) B, const size\_t ldb, int \*info)  
*Solve the system  $AX = B$  or  $XA = B$ .*

## 13.192.1 Macro Definition Documentation

### 13.192.1.1 \_\_FFLASFFPACK\_ffpack\_fgetrs\_INL

```
#define __FFLASFFPACK_ffpack_fgetrs_INL
```

## 13.193 ffpack\_frobenius.inl File Reference

```
#include <givaro/givranditer.h>
```

## Namespaces

- namespace [FFPACK](#)  
*Finite **Field** **PACK** Set of elimination based routines for dense linear algebra.*
- namespace [FFPACK::Protected](#)

## Functions

- template<class [Field](#) >  
void [CompressRows](#) ([Field](#) &F, const size\_t M, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) tmp, const size\_t ldtmp, const size\_t \*d, const size\_t nb\_blocs)
- template<class [Field](#) >  
void [CompressRowsQK](#) ([Field](#) &F, const size\_t M, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) tmp, const size\_t ldtmp, const size\_t \*d, const size\_t deg, const size\_t nb\_blocs)
- template<class [Field](#) >  
void [DeCompressRows](#) ([Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) tmp, const size\_t ldtmp, const size\_t \*d, const size\_t nb\_blocs)
- template<class [Field](#) >  
void [DeCompressRowsQK](#) ([Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) tmp, const size\_t ldtmp, const size\_t \*d, const size\_t deg, const size\_t nb\_blocs)
- template<class [Field](#) >  
void [CompressRowsQA](#) ([Field](#) &F, const size\_t M, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) tmp, const size\_t ldtmp, const size\_t \*d, const size\_t nb\_blocs)
- template<class [Field](#) >  
void [DeCompressRowsQA](#) ([Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) tmp, const size\_t ldtmp, const size\_t \*d, const size\_t nb\_blocs)
- template<class [PolRing](#) >  
void [RandomKrylovPrecond](#) (const [PolRing](#) &PR, std::list< typename [PolRing::Element](#) > &completedFactors, const size\_t N, typename [PolRing::Domain\\_t::Element\\_ptr](#) A, const size\_t lda, size\_t &Nb, typename [PolRing::Domain\\_t::Element\\_ptr](#) &B, size\_t &ldb, typename [PolRing::Domain\\_t::RandIter](#) &g, const size\_t degree=\_\_FFLASFFPACK\_ARITHPROG\_THRESHOLD)

- `template<class PolRing >`  
`std::list< typename PolRing::Element > & ArithProg (const PolRing &PR, std::list< typename PolRing::Element > &frobeniusForm, const size_t N, typename PolRing::Domain_t::Element_ptr A, const size_t lda, const size_t degree)`

## 13.194 ffpack\_fsytrf.inl File Reference

```
#include "fflas-ffpack/utils/fflas_io.h"
```

### Namespaces

- namespace `FFPACK`  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

### Macros

- `#define __FFLASFFPACK_ffpack_fsytrf_INL`

### Functions

- `template<class Field >`  
`bool fsytrf_BC_Crout (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr Dinv, const size_t incDinv)`
- `template<class Field >`  
`size_t fsytrf_BC_RL (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr Dinv, const size_t incDinv)`
- `template<class Field >`  
`size_t fsytrf_UP_RPM_BC_RL (const Field &F, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr Dinv, const size_t incDinv, size_t *P)`
- `template<class Field >`  
`size_t fsytrf_LOW_RPM_BC_Crout (const Field &F, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr Dinv, const size_t incDinv, size_t *P)`
- `template<class Field >`  
`size_t fsytrf_UP_RPM_BC_Crout (const Field &F, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr Dinv, const size_t incDinv, size_t *P)`
- `template<class Field >`  
`size_t fsytrf_UP_RPM (const Field &Fi, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr Dinv, const size_t incDinv, size_t *P, size_t BCThreshold)`
- `template<class Field >`  
`bool fsytrf_nonunit (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr Dinv, const size_t incDinv, FFLAS::ParSeqHelper::Sequential seq, size_t threshold)`
- `template<class Field, class Cut, class Param >`  
`bool fsytrf_nonunit (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t lda, typename Field::Element_ptr Dinv, const size_t incDinv, FFLAS::ParSeqHelper::Parallel< Cut, Param > par, size_t threshold)`
- `template<class Field >`  
`bool fsytrf (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t lda, const size_t threshold=__FFLASFFPACK_FSYTRF_THRESHOLD)`  
*Triangular factorization of symmetric matrices.*
- `template<class Field >`  
`bool fsytrf (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t lda, const FFLAS::ParSeqHelper::Sequential seq, const size_t threshold=__FFLASFFPACK_FSYTRF_THRESHOLD)`
- `template<class Field, class Cut, class Param >`  
`bool fsytrf (const Field &F, const FFLAS::FFLAS_UPLO UpLo, const size_t N, typename Field::Element_ptr A, const size_t lda, const FFLAS::ParSeqHelper::Parallel< Cut, Param > par, const size_t threshold=__FFLASFFPACK_FSYTRF_THRESHOLD)`

- template<class [Field](#) >  
size\_t [fsytrf\\_RPM](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_UPLO](#) UpLo, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*P, size\_t threshold)
- template<class [Field](#) >  
void [getTridiagonal](#) (const [Field](#) &F, const size\_t N, const size\_t R, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, size\_t \*P, typename [Field::Element\\_ptr](#) T, const size\_t ldt)

### 13.194.1 Macro Definition Documentation

#### 13.194.1.1 \_\_FFLASFFPACK\_ffpack\_fsytrf\_INL

```
#define __FFLASFFPACK_ffpack_fsytrf_INL
```

## 13.195 ffpack\_ftrssyr2k.inl File Reference

```
#include "fflas-ffpack/utils/fflas_io.h"
```

### Namespaces

- namespace [FFPACK](#)  
*Finite **Field** **PACK** Set of elimination based routines for dense linear algebra.*

### Macros

- #define [\\_\\_FFLASFFPACK\\_ffpack\\_ftrssyr2k\\_INL](#)

### Functions

- template<class [Field](#) >  
void [ftrssyr2k](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_UPLO](#) Uplo, const [FFLAS::FFLAS\\_DIAG](#) diagA, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) B, const size\_t ldb, const size\_t threshold=[\\_\\_FFLASFFPACK\\_FTRSSYR2K\\_THRESHOLD](#))  
*Solve a triangular system in a symmetric sum: find B upper/lower triangular such that  $A^T B + B^T A = C$  where C is symmetric.*

### 13.195.1 Macro Definition Documentation

#### 13.195.1.1 \_\_FFLASFFPACK\_ffpack\_ftrssyr2k\_INL

```
#define __FFLASFFPACK_ffpack_ftrssyr2k_INL
```

## 13.196 ffpack\_ftrstr.inl File Reference

```
#include "fflas-ffpack/utils/fflas_io.h"
```

### Namespaces

- namespace [FFPACK](#)  
*Finite **Field** **PACK** Set of elimination based routines for dense linear algebra.*

### Macros

- #define [\\_\\_FFLASFFPACK\\_ffpack\\_ftrstr\\_INL](#)

## Functions

- `template<class Field >`  
`void ftrstr (const Field &F, const FFLAS::FFLAS_SIDE side, const FFLAS::FFLAS_UPLO Uplo,`  
`const FFLAS::FFLAS_DIAG diagA, const FFLAS::FFLAS_DIAG diagB, const size_t N, typename`  
`Field::ConstElement_ptr A, const size_t lda, typename Field::Element_ptr B, const size_t ldb, const size_t`  
`threshold=__FFLASFFPACK_FTRSTR_THRESHOLD)`

*Solve a triangular system with a triangular right hand side of the same shape.*

## 13.196.1 Macro Definition Documentation

### 13.196.1.1 \_\_FFLASFFPACK\_ffpack\_ftrstr\_INL

```
#define __FFLASFFPACK_ffpack_ftrstr_INL
```

## 13.197 fpack\_fttr.inl File Reference

### Namespaces

- namespace `FFPACK`  
*Finite Field PACK Set of elimination based routines for dense linear algebra.*

### Macros

- `#define ENABLE_ALL_CHECKINGS 1`
- `#define __FFLASFFPACK_ffpack_fttr_INL`

## Functions

- `template<class Field >`  
`void ftrtri (const Field &F, const FFLAS::FFLAS_UPLO Uplo, const FFLAS::FFLAS_DIAG Diag, const size_t`  
`N, typename Field::Element_ptr A, const size_t lda, const size_t threshold=__FFLASFFPACK_FTRTRI_THRESHOLD)`  
*Compute the inverse of a triangular matrix.*
- `template<class Field >`  
`void ftrtrm (const Field &F, const FFLAS::FFLAS_SIDE side, const FFLAS::FFLAS_DIAG diag, const size_t`  
`N, typename Field::Element_ptr A, const size_t lda)`  
*Compute the product of two triangular matrices of opposite shape.*
- `template<class Field >`  
`void trinv_left (const Field &F, const size_t N, typename Field::ConstElement_ptr L, const size_t ldl, typename`  
`Field::Element_ptr X, const size_t idx)`

## 13.197.1 Macro Definition Documentation

### 13.197.1.1 ENABLE\_ALL\_CHECKINGS

```
#define ENABLE_ALL_CHECKINGS 1
```

### 13.197.1.2 \_\_FFLASFFPACK\_ffpack\_fttr\_INL

```
#define __FFLASFFPACK_ffpack_fttr_INL
```

## 13.198 fpack\_invert.inl File Reference

### Namespaces

- namespace `FFPACK`  
*Finite Field PACK Set of elimination based routines for dense linear algebra.*

**Macros**

- `#define __FFLASFFPACK_ffpack_invert_INL`

**Functions**

- `template<class Field >`  
`Field::Element_ptr Invert` (const `Field` &F, const `size_t` M, typename `Field::Element_ptr` A, const `size_t` lda, int &>nullity)  
*Invert the given matrix in place or computes its nullity if it is singular.*
- `template<class Field >`  
`Field::Element_ptr Invert` (const `Field` &F, const `size_t` M, typename `Field::ConstElement_ptr` A, const `size_t` lda, typename `Field::Element_ptr` X, const `size_t` ldx, int &>nullity)  
*Invert the given matrix or computes its nullity if it is singular.*
- `template<class Field >`  
`Field::Element_ptr Invert2` (const `Field` &F, const `size_t` M, typename `Field::Element_ptr` A, const `size_t` lda, typename `Field::Element_ptr` X, const `size_t` ldx, int &>nullity)  
*Invert the given matrix or computes its nullity if it is singular.*

**13.198.1 Macro Definition Documentation****13.198.1.1 \_\_FFLASFFPACK\_ffpack\_invert\_INL**

```
#define __FFLASFFPACK_ffpack_invert_INL
```

**13.199 ffpack\_krylovelim.inl File Reference****Macros**

- `#define __FFLASFFPACK_ffpack_krylovelim_INL`

**13.199.1 Macro Definition Documentation****13.199.1.1 \_\_FFLASFFPACK\_ffpack\_krylovelim\_INL**

```
#define __FFLASFFPACK_ffpack_krylovelim_INL
```

**13.200 ffpack\_ludivine.inl File Reference**

```
#include "fflas-ffpack/fflas/fflas_bounds.inl"
```

**Data Structures**

- class `callLUdivine_small< Element >`
- class `callLUdivine_small< double >`
- class `callLUdivine_small< float >`

**Namespaces**

- namespace `FFPACK`  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*
- namespace `FFPACK::Protected`

**Macros**

- `#define __FFLASFFPACK_ffpack_ludivine_INL`

## Functions

- `template<class Field >`  
`size_t LUdivine_gauss` (const `Field` &F, const `FFLAS::FFLAS_DIAG` Diag, const `size_t` M, const `size_t` N, typename `Field::Element_ptr` A, const `size_t` lda, `size_t` \*P, `size_t` \*Q, const `FFPACK::FFPACK_LU_TAG` LuTag)
- `template<class Field >`  
`size_t LUdivine_small` (const `Field` &F, const `FFLAS::FFLAS_DIAG` Diag, const `FFLAS::FFLAS_TRANSPOSE` trans, const `size_t` M, const `size_t` N, typename `Field::Element_ptr` A, const `size_t` lda, `size_t` \*P, `size_t` \*Q, const `FFPACK::FFPACK_LU_TAG` LuTag)
- `template<class Field >`  
`size_t LUdivine` (const `Field` &F, const `FFLAS::FFLAS_DIAG` Diag, const `FFLAS::FFLAS_TRANSPOSE` trans, const `size_t` M, const `size_t` N, typename `Field::Element_ptr` A, const `size_t` lda, `size_t` \*P, `size_t` \*Q, const `FFPACK::FFPACK_LU_TAG` LuTag, const `size_t` cutoff)
- `template<class Field >`  
`size_t LUdivine_construct` (const `Field` &F, const `FFLAS::FFLAS_DIAG` Diag, const `size_t` M, const `size_t` N, typename `Field::ConstElement_ptr` A, const `size_t` lda, typename `Field::Element_ptr` X, const `size_t` idx, typename `Field::Element_ptr` u, const `size_t` incu, `size_t` \*P, bool computeX, const `FFPACK::FFPACK_MINPOLY_TAG` MinTag, const `size_t` kg\_mc, const `size_t` kg\_mb, const `size_t` kg\_j)

### 13.200.1 Macro Definition Documentation

#### 13.200.1.1 \_\_FFLASFFPACK\_ffpack\_ludivine\_INL

```
#define __FFLASFFPACK_ffpack_ludivine_INL
```

## 13.201 ffpack\_ludivine\_mp.inl File Reference

```
#include <givaro/modular-integer.h>
#include <givaro/givinteger.h>
#include "fflas-ffpack/field/rns-integer-mod.h"
#include "fflas-ffpack/field/rns-integer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/ffpack/ffpack_ludivine.inl"
```

## Namespaces

- namespace `FFPACK`  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

## Macros

- `#define __FFPACK_ludivine_mp_INL`

## Functions

- `template<> size_t LUdivine` (const `Givaro::Modular< Givaro::Integer >` &F, const `FFLAS::FFLAS_DIAG` Diag, const `FFLAS::FFLAS_TRANSPOSE` trans, const `size_t` M, const `size_t` N, typename `Givaro::Integer` \*A, const `size_t` lda, `size_t` \*P, `size_t` \*Q, const `FFPACK::FFPACK_LU_TAG` LuTag, const `size_t` cutoff)

### 13.201.1 Macro Definition Documentation

#### 13.201.1.1 \_\_FFPACK\_ludivine\_mp\_INL

```
#define __FFPACK_ludivine_mp_INL
```

## 13.202 ffpack\_minpoly.inl File Reference

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*
- namespace [FFPACK::Protected](#)

### Macros

- #define [\\_\\_FFLASFFPACK\\_ffpack\\_minpoly\\_INL](#)

### Functions

- template<class [Field](#) , class Polynomial >  
Polynomial & [MinPoly](#) (const [Field](#) &F, Polynomial &minP, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda)  
*Compute the minimal polynomial of the matrix A.*
- template<class [Field](#) , class Polynomial , class RandIter >  
Polynomial & [MinPoly](#) (const [Field](#) &F, Polynomial &minP, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, RandIter &G)  
*Compute the minimal polynomial of the matrix A.*
- template<class [Field](#) , class Polynomial >  
Polynomial & [MatVecMinPoly](#) (const [Field](#) &F, Polynomial &minP, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::ConstElement\\_ptr](#) v, const size\_t incv)  
*Compute the minimal polynomial of the matrix A and a vector v, namely the first linear dependency relation in the Krylov basis  $(v, Av, \dots, A^N v)$ .*
- template<class [Field](#) , class Polynomial >  
Polynomial & [MatVecMinPoly](#) (const [Field](#) &F, Polynomial &minP, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) v, const size\_t incv, typename [Field::Element\\_ptr](#) K, const size\_t ldk, size\_t \*P)  
*Compute the minimal polynomial of the matrix A and a vector v, namely the first linear dependency relation in the Krylov basis  $(v, Av, \dots, A^N v)$ .*
- template<class [Field](#) , class Polynomial >  
Polynomial & [Hybrid\\_KGF\\_LUK\\_MinPoly](#) (const [Field](#) &F, Polynomial &minP, const size\_t N, typename [Field::ConstElement\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) X, const size\_t ldx, size\_t \*P, const FFPACK\_MINPOLY\_TAG MinTag=FFPACK::FfpackDense, const size\_t kg\_mc=0, const size\_t kg\_↔ mb=0, const size\_t kg\_j=0)

### 13.202.1 Macro Definition Documentation

#### 13.202.1.1 \_\_FFLASFFPACK\_ffpack\_minpoly\_INL

```
#define __FFLASFFPACK_ffpack_minpoly_INL
```

## 13.203 ffpack\_permutation.inl File Reference

```
#include <givar/zring.h>
#include "fflas-ffpack/fflas/fflas_fassign.h"
```

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

### Macros

- #define [\\_\\_FFLASFFPACK\\_ffpack\\_permutation\\_INL](#)
- #define [FFLASFFPACK\\_PERM\\_BKSIZE](#) 32

## Functions

- template<class [Field](#) >  
void [MonotonicApplyP](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const [FFLAS::FFLAS\\_TRANSPOSE](#) Trans, const size\_t M, const size\_t ibeg, const size\_t iend, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t \*P, const size\_t R)  
*Apply a R-monotonically increasing permutation P, to the matrix A.*
- template<class [Field](#) >  
void [MonotonicCompress](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t incA, const size\_t \*MathP, const size\_t R, const size\_t maxpiv, const size\_t rowstomove, const std::vector< bool > &ispiv)
- template<class [Field](#) >  
void [MonotonicCompressMorePivots](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, type-name [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t incA, const size\_t \*MathP, const size\_t R, const size\_t rowstomove, const size\_t lenP)
- template<class [Field](#) >  
void [MonotonicCompressCycles](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t incA, const size\_t \*MathP, const size\_t lenP)
- template<class [Field](#) >  
void [MonotonicExpand](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t incA, const size\_t \*MathP, const size\_t R, const size\_t maxpiv, const size\_t rowstomove, const std::vector< bool > &ispiv)
- template<class [Field](#) >  
void [applyP\\_block](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const [FFLAS::FFLAS\\_TRANSPOSE](#) Trans, const size\_t M, const size\_t ibeg, const size\_t iend, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t \*P)
- template<class [Field](#) >  
void [applyP](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const [FFLAS::FFLAS\\_TRANSPOSE](#) Trans, const size\_t m, const size\_t ibeg, const size\_t iend, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t \*P, const [FFLAS::ParSeqHelper::Sequential](#) seq)
- template<class [Field](#) >  
void [doApplyS](#) (const [Field](#) &F, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) tmp, const size\_t width, const size\_t M2, const size\_t R1, const size\_t R2, const size\_t R3, const size\_t R4)
- template<class [Field](#) >  
void [MatrixApplyS](#) (const [Field](#) &F, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t width, const size\_t M2, const size\_t R1, const size\_t R2, const size\_t R3, const size\_t R4)
- template<class [Field](#) >  
void [MatrixApplyS](#) (const [Field](#) &F, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t width, const size\_t M2, const size\_t R1, const size\_t R2, const size\_t R3, const size\_t R4, const [FFLAS::ParSeqHelper::Sequential](#) seq)
- template<class [Field](#) , class Cut , class Param >  
void [MatrixApplyS](#) (const [Field](#) &F, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t width, const size\_t M2, const size\_t R1, const size\_t R2, const size\_t R3, const size\_t R4, const [FFLAS::ParSeqHelper::Parallel](#)< Cut, Param > par)
- template<class T >  
void [PermApplyS](#) (T \*A, const size\_t lda, const size\_t width, const size\_t M2, const size\_t R1, const size\_t R2, const size\_t R3, const size\_t R4)
- template<class [Field](#) >  
void [doApplyT](#) (const [Field](#) &F, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) tmp, const size\_t width, const size\_t N2, const size\_t R1, const size\_t R2, const size\_t R3, const size\_t R4)
- template<class [Field](#) >  
void [MatrixApplyT](#) (const [Field](#) &F, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t width, const size\_t N2, const size\_t R1, const size\_t R2, const size\_t R3, const size\_t R4)
- template<class [Field](#) >  
void [MatrixApplyT](#) (const [Field](#) &F, typename [Field::Element\\_ptr](#) A, const size\_t lda, const size\_t width, const size\_t N2, const size\_t R1, const size\_t R2, const size\_t R3, const size\_t R4, const [FFLAS::ParSeqHelper::Sequential](#) seq)



- `template<class Field , class Cut , class Param >`  
`void MatrixApplyT (const Field &F, typename Field::Element_ptr A, const size_t lda, const size_t width, const size_t N2, const size_t R1, const size_t R2, const size_t R3, const size_t R4, const FFLAS::ParSeqHelper::Parallel< Cut, Param > par)`
- `template<class T >`  
`void PermApplyT (T *A, const size_t lda, const size_t width, const size_t N2, const size_t R1, const size_t R2, const size_t R3, const size_t R4)`
- `void LAPACKPerm2MathPerm (size_t *MathP, const size_t *LapackP, const size_t N)`  
*Conversion of a permutation from LAPACK format to Math format.*
- `void MathPerm2LAPACKPerm (size_t *LapackP, const size_t *MathP, const size_t N)`  
*Conversion of a permutation from Maths format to LAPACK format.*
- `void composePermutationsLLL (size_t *P1, const size_t *P2, const size_t R, const size_t N)`  
*Computes  $P1 \times \text{Diag}(I_R, P2)$  where  $P1$  is a LAPACK and  $P2$  a LAPACK permutation and store the result in  $P1$  as a LAPACK permutation.*
- `void composePermutationsLLM (size_t *MathP, const size_t *P1, const size_t *P2, const size_t R, const size_t N)`  
*Computes  $P1 \times \text{Diag}(I_R, P2)$  where  $P1$  is a LAPACK and  $P2$  a LAPACK permutation and store the result in  $MathP$  as a MathPermutation format.*
- `void composePermutationsMLM (size_t *MathP1, const size_t *P2, const size_t R, const size_t N)`  
*Computes  $MathP1 \times \text{Diag}(I_R, P2)$  where  $MathP1$  is a MathPermutation and  $P2$  a LAPACK permutation and store the result in  $MathP1$  as a MathPermutation format.*
- `void cyclic_shift_mathPerm (size_t *P, const size_t s)`
- `template<class Field >`  
`void cyclic_shift_row_col (const Field &F, typename Field::Element_ptr A, size_t m, size_t n, size_t lda)`
- `template<class Field >`  
`void cyclic_shift_row (const Field &F, typename Field::Element_ptr A, size_t m, size_t n, size_t lda)`
- `template<typename T >`  
`void cyclic_shift_row (const RNSIntegerMod< T > &F, typename T::Element_ptr A, size_t m, size_t n, size_t lda)`
- `template<class Field >`  
`void cyclic_shift_col (const Field &F, typename Field::Element_ptr A, size_t m, size_t n, size_t lda)`
- `template<typename T >`  
`void cyclic_shift_col (const RNSIntegerMod< T > &F, typename T::Element_ptr A, size_t m, size_t n, size_t lda)`
- `template<class Field >`  
`void applyP (const Field &F, const FFLAS::FFLAS_SIDE Side, const FFLAS::FFLAS_TRANSPOSE Trans, const size_t M, const size_t ibeg, const size_t iend, typename Field::Element_ptr A, const size_t lda, const size_t *P)`  
*Computes  $P1 \times \text{Diag}(I_R, P2)$  where  $P1$  is a LAPACK and  $P2$  a LAPACK permutation and store the result in  $P1$  as a LAPACK permutation.*
- `template<class Field , class Cut , class Param >`  
`void applyP (const Field &F, const FFLAS::FFLAS_SIDE Side, const FFLAS::FFLAS_TRANSPOSE Trans, const size_t m, const size_t ibeg, const size_t iend, typename Field::Element_ptr A, const size_t lda, const size_t *P, const FFLAS::ParSeqHelper::Parallel< Cut, Param > par)`

## 13.203.1 Macro Definition Documentation

### 13.203.1.1 \_\_FFLASFFPACK\_ffpack\_permutation\_INL

```
#define __FFLASFFPACK_ffpack_permutation_INL
```

### 13.203.1.2 FFLASFFPACK\_PERM\_BKSIZE

```
#define FFLASFFPACK_PERM_BKSIZE 32
```

## 13.204 ffpack\_pluq.inl File Reference

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

### Macros

- #define [\\_\\_FFLASFFPACK\\_ffpack\\_pluq\\_INL](#)
- #define [CROUT](#)

### Functions

- template<class [Field](#) >  
size\_t [PLUQ\\_basecaseV3](#) (const [Field](#) &Fi, const [FFLAS::FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, typename [Field::Element](#) \*A, const size\_t lda, size\_t \*P, size\_t \*Q)
- template<class [Field](#) >  
size\_t [PLUQ\\_basecaseV2](#) (const [Field](#) &Fi, const [FFLAS::FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, typename [Field::Element](#) \*A, const size\_t lda, size\_t \*P, size\_t \*Q)
- template<class [Field](#) >  
size\_t [PLUQ\\_basecaseCROUT](#) (const [Field](#) &Fi, const [FFLAS::FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*P, size\_t \*Q)
- template<class [Field](#) >  
size\_t [\\_PLUQ](#) (const [Field](#) &Fi, const [FFLAS::FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*P, size\_t \*Q, size\_t BCThreshold)
- template<class [Field](#) >  
size\_t [PLUQ](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*P, size\_t \*Q, const [FFLAS::ParSeqHelper::Sequential](#) &PShelper, size\_t BCThreshold=[\\_\\_FFLASFFPACK\\_PLUQ\\_THRESHOLD](#))
- template<class [Field](#) >  
size\_t [PLUQ](#) (const [Field](#) &F, const [FFLAS::FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*P, size\_t \*Q)

*Compute a PLUQ factorization of the given matrix.*

### 13.204.1 Macro Definition Documentation

#### 13.204.1.1 [\\_\\_FFLASFFPACK\\_ffpack\\_pluq\\_INL](#)

```
#define __FFLASFFPACK_ffpack_pluq_INL
```

#### 13.204.1.2 [CROUT](#)

```
#define CROUT
```

## 13.205 ffpack\_pluq\_mp.inl File Reference

```
#include "fflas-ffpack/field/rns-integer-mod.h"
#include "fflas-ffpack/field/rns-integer.h"
#include "fflas-ffpack/fflas-ffpack.h"
#include "givaro/givinteger.h"
#include "givaro/modular-integer.h"
```

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

## Macros

- `#define __FFPACK_pluq_mp_INL`

## Functions

- `template<class Cut , class Param >`  
`size_t PLUQ (const Givaro::Modular< Givaro::Integer > &F, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Givaro::Integer *A, const size_t lda, size_t *P, size_t *Q, size_t BCThreshold, FFLAS::ParSeqHelper::Parallel< Cut, Param > &PSHelper)`

## 13.205.1 Macro Definition Documentation

### 13.205.1.1 \_\_FFPACK\_pluq\_mp\_INL

```
#define __FFPACK_pluq_mp_INL
```

## 13.206 ffpack\_ppluq.inl File Reference

## Namespaces

- namespace `FFPACK`  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

## Macros

- `#define __FFLASFFPACK_ffpack_ppluq_INL`
- `#define __FFLAS__TRSM_READONLY`
- `#define PBASECASE_K 256`

## Functions

- `template<class Field >`  
`void threads_fgemm (const size_t m, const size_t n, const size_t r, int nbthreads, size_t *W1, size_t *W2, size_t *W3, size_t gamma)`
- `template<class Field >`  
`void threads_ftsm (const size_t m, const size_t n, int nbthreads, size_t *t1, size_t *t2)`
- `template<class Field >`  
`size_t PLUQ (const Field &Fi, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q, const FFLAS::ParSeqHelper::Parallel< FFLAS::CuttingStrategy::Recursive, FFLAS::StrategyParameter::Threads > &PSHelper)`
- `template<class Field >`  
`size_t pPLUQ (const Field &F, const FFLAS::FFLAS_DIAG Diag, const size_t M, const size_t N, typename Field::Element_ptr A, const size_t lda, size_t *P, size_t *Q)`

## 13.206.1 Macro Definition Documentation

### 13.206.1.1 \_\_FFLASFFPACK\_ffpack\_ppluq\_INL

```
#define __FFLASFFPACK_ffpack_ppluq_INL
```

### 13.206.1.2 \_\_FFLAS\_\_TRSM\_READONLY

```
#define __FFLAS__TRSM_READONLY
```

### 13.206.1.3 PBASECASE\_K

```
#define PBASECASE_K 256
```

## 13.207 ffpack\_rankprofiles.inl File Reference

### Namespaces

- namespace [FFPACK](#)  
*Finite Field PACK Set of elimination based routines for dense linear algebra.*

### Macros

- #define [\\_\\_FFLASFFPACK\\_ffpack\\_rank\\_profiles\\_INL](#)

### Functions

- template<class [Field](#) >  
size\_t [RowRankProfile](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*rkprofile, const FFPACK\_LU\_TAG LuTag=[FfpackSlabRecursive](#))  
*Computes the row rank profile of A.*
- template<class [Field](#) >  
size\_t [pRowRankProfile](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*rkprofile, size\_t numthreads=0, const FFPACK\_LU\_TAG LuTag=[FfpackTileRecursive](#))
- template<class [Field](#), class PSHelper >  
size\_t [RowRankProfile](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*rkprofile, const FFPACK\_LU\_TAG LuTag, PSHelper &psH)
- template<class [Field](#) >  
size\_t [ColumnRankProfile](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*rkprofile, const FFPACK\_LU\_TAG LuTag=[FfpackSlabRecursive](#))  
*Computes the column rank profile of A.*
- template<class [Field](#) >  
size\_t [pColumnRankProfile](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*rkprofile, size\_t numthreads=0, const FFPACK\_LU\_TAG LuTag=[FfpackTileRecursive](#))
- template<class [Field](#), class PSHelper >  
size\_t [ColumnRankProfile](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*rkprofile, const FFPACK\_LU\_TAG LuTag, PSHelper &psH)
- void [RankProfileFromLU](#) (const size\_t \*P, const size\_t N, const size\_t R, size\_t \*rkprofile, const FFPACK\_LU\_TAG LuTag)  
*Recovers the column/row rank profile from the permutation of an LU decomposition.*
- size\_t [LeadingSubmatrixRankProfiles](#) (const size\_t M, const size\_t N, const size\_t R, const size\_t LSm, const size\_t LSn, const size\_t \*P, const size\_t \*Q, size\_t \*RRP, size\_t \*CRP)  
*Recovers the row and column rank profiles of any leading submatrix from the PLUQ decomposition.*
- template<class [Field](#) >  
size\_t [RowRankProfileSubmatrixIndices](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*rowindices, size\_t \*colindices, size\_t &R)  
*RowRankProfileSubmatrixIndices.*
- template<class [Field](#) >  
size\_t [ColRankProfileSubmatrixIndices](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, size\_t \*rowindices, size\_t \*colindices, size\_t &R)  
*Computes the indices of the submatrix  $r \times r$  X of A whose columns correspond to the column rank profile of A.*
- template<class [Field](#) >  
size\_t [RowRankProfileSubmatrix](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) &X, size\_t &R)  
*Computes the  $r \times r$  submatrix X of A, by picking the row rank profile rows of A.*
- template<class [Field](#) >  
size\_t [ColRankProfileSubmatrix](#) (const [Field](#) &F, const size\_t M, const size\_t N, typename [Field::Element\\_ptr](#) A, const size\_t lda, typename [Field::Element\\_ptr](#) &X, size\_t &R)  
*Compute the  $r \times r$  submatrix X of A, by picking the row rank profile rows of A.*

- `template<class Field >`  
`Field::Element_ptr LQUPtoInverseOfFullRankMinor (const Field &F, const size_t rank, typename`  
`Field::Element_ptr A_factors, const size_t lda, const size_t *QtPointer, typename Field::Element_ptr X,`  
`const size_t ldx)`  
`LQUPtoInverseOfFullRankMinor.`

## 13.207.1 Macro Definition Documentation

### 13.207.1.1 \_\_FFLASFFPACK\_ffpack\_rank\_profiles\_INL

```
#define __FFLASFFPACK_ffpack_rank_profiles_INL
```

## 13.208 field-traits.h File Reference

Field Traits.

```
#include <type_traits>
#include "fflas-ffpack/field/rns-double-elt.h"
#include "recint/rmint.h"
#include "givaro/modular-general.h"
#include "givaro/zring.h"
```

### Data Structures

- struct [GenericTag](#)  
*generic ring.*
- struct [ModularTag](#)  
*This is a modular field like e.g. Modular<T> or ModularBalanced<T>*
- struct [UnparametricTag](#)  
*If the field uses a representation with infix operators.*
- struct [DefaultTag](#)  
*No specific mode of action: use standard field operations.*
- struct [DefaultBoundedTag](#)  
*Use standard field operations, but keeps track of bounds on input and output.*
- struct [ConvertTo< T >](#)  
*Force conversion to appropriate element type of ElementCategory T.*
- struct [DelayedTag](#)  
*Performs field operations with delayed mod reductions. Ensures result is reduced.*
- struct [LazyTag](#)  
*Performs field operations with delayed mod only when necessary. Result may not be reduced.*
- struct [GenericTag](#)  
*default is generic*
- struct [MachineFloatTag](#)  
*float or double*
- struct [MachineIntTag](#)  
*short, int, long, long long, and unsigned variants*
- struct [FixedPrecIntTag](#)  
*Fixed precision integers above machine precision: Givaro::recInt.*
- struct [ArbitraryPrecIntTag](#)  
*Arbitrary precision integers: GMP.*
- struct [RNSElementTag](#)  
*Representation in a Residue Number System.*
- struct [ElementTraits< Element >](#)

*ElementTraits.*

- struct `ElementTraits< float >`
- struct `ElementTraits< double >`
- struct `ElementTraits< int8_t >`
- struct `ElementTraits< int16_t >`
- struct `ElementTraits< int32_t >`
- struct `ElementTraits< int64_t >`
- struct `ElementTraits< uint8_t >`
- struct `ElementTraits< uint16_t >`
- struct `ElementTraits< uint32_t >`
- struct `ElementTraits< uint64_t >`
- struct `ElementTraits< Givaro::Integer >`
- struct `ElementTraits< Reclnt::rint< K > >`
- struct `ElementTraits< Reclnt::ruint< K > >`
- struct `ElementTraits< Reclnt::rmint< K, MG > >`
- struct `ElementTraits< FFPACK::rns_double_elt >`
- struct `ModeTraits< Field >`

*ModeTraits.*

- struct `ModeTraits< Givaro::Modular< Element, Compute > >`
- struct `ModeTraits< Givaro::Modular< int64_t, uint64_t > >`
- struct `ModeTraits< Givaro::Modular< int8_t, Compute > >`
- struct `ModeTraits< Givaro::Modular< int16_t, Compute > >`
- struct `ModeTraits< Givaro::Modular< int32_t, Compute > >`
- struct `ModeTraits< Givaro::Modular< uint8_t, Compute > >`
- struct `ModeTraits< Givaro::Modular< uint16_t, Compute > >`
- struct `ModeTraits< Givaro::Modular< uint32_t, Compute > >`
- struct `ModeTraits< Givaro::Modular< Givaro::Integer, Compute > >`
- struct `ModeTraits< Givaro::Modular< Reclnt::ruint< K >, Compute > >`
- struct `ModeTraits< Givaro::ModularBalanced< Element > >`
- struct `ModeTraits< Givaro::ModularBalanced< int8_t > >`
- struct `ModeTraits< Givaro::ModularBalanced< int16_t > >`
- struct `ModeTraits< Givaro::ModularBalanced< int32_t > >`
- struct `ModeTraits< Givaro::ModularBalanced< Givaro::Integer > >`
- struct `ModeTraits< Givaro::ZRing< Givaro::Integer > >`
- struct `ModeTraits< Givaro::ZRing< float > >`
- struct `ModeTraits< Givaro::ZRing< double > >`
- struct `ModeTraits< Givaro::Montgomery< T > >`
- struct `FieldTraits< Field >`

*FieldTrait.*

- struct `FieldTraits< Givaro::ZRing< Reclnt::ruint< K > > >`
- struct `FieldTraits< Givaro::Modular< Element > >`
- struct `FieldTraits< Givaro::ModularBalanced< Element > >`
- struct `FieldTraits< Givaro::ZRing< double > >`
- struct `FieldTraits< Givaro::ZRing< float > >`
- struct `FieldTraits< Givaro::ZRing< int16_t > >`
- struct `FieldTraits< Givaro::ZRing< uint16_t > >`
- struct `FieldTraits< Givaro::ZRing< int32_t > >`
- struct `FieldTraits< Givaro::ZRing< uint32_t > >`
- struct `FieldTraits< Givaro::ZRing< int64_t > >`
- struct `FieldTraits< Givaro::ZRing< uint64_t > >`
- struct `FieldTraits< Givaro::ZRing< Givaro::Integer > >`
- struct `FieldTraits< FFPACK::RNSInteger< T > >`
- struct `FieldTraits< FFPACK::RNSIntegerMod< T > >`
- struct `associatedDelayedField< Field >`
- struct `associatedDelayedField< const Givaro::Modular< T, X > >`
- struct `associatedDelayedField< const Givaro::ModularBalanced< T > >`
- struct `associatedDelayedField< const Givaro::ZRing< T > >`
- struct `associatedDelayedField< const FFPACK::RNSIntegerMod< RNS > >`

## Namespaces

- namespace [Reclnt](#)
- namespace [Givaro](#)
- namespace [FFPACK](#)

*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

- namespace [FFLAS](#)
- namespace [FFLAS::FieldCategories](#)

*Traits and categories will need to be placed in a proper file later.*

- namespace [FFLAS::ModeCategories](#)
- namespace [FFLAS::ElementCategories](#)

*Specifies the mode of action for an algorithm w.r.t.*

## Functions

- template<class [Field](#) , class enable = void>  
Field::Residu\_t [maxCardinality](#) ()
- template<> uint64\_t [maxCardinality](#)< [Givaro::Modular](#)< int64\_t > > ()
- template<> uint32\_t [maxCardinality](#)< [Givaro::Modular](#)< int32\_t > > ()
- template<class [Field](#) >  
Field::Residu\_t [minCardinality](#) ()

### 13.208.1 Detailed Description

Field Traits.

## 13.209 field.doxy File Reference

### 13.210 rns-double-elt.h File Reference

rns elt structure with double support

```
#include "fflas-ffpack/utils/fflas_memory.h"
#include "fflas-ffpack/utils/cast.h"
```

## Data Structures

- struct [rns\\_double\\_elt](#)
- struct [rns\\_double\\_elt\\_ptr](#)
- struct [rns\\_double\\_elt\\_cstptr](#)

## Namespaces

- namespace [FFPACK](#)

*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

## Functions

- template<> [rns\\_double\\_elt\\_ptr](#) fflas\_const\_cast ([rns\\_double\\_elt\\_cstptr](#) x)
- template<> [rns\\_double\\_elt\\_cstptr](#) fflas\_const\_cast ([rns\\_double\\_elt\\_ptr](#) x)

### 13.210.1 Detailed Description

rns elt structure with double support

## 13.211 rns-double-recint.inl File Reference

```
#include "fflas-ffpack/fflas/fflas_freduce.h"
```

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

### Macros

- #define [\\_\\_FFLASFFPACK\\_field\\_rns\\_double\\_recint\\_INL](#)

### 13.211.1 Macro Definition Documentation

#### 13.211.1.1 [\\_\\_FFLASFFPACK\\_field\\_rns\\_double\\_recint\\_INL](#)

```
#define __FFLASFFPACK_field_rns_double_recint_INL
```

## 13.212 rns-double.h File Reference

rns structure with double support

```
#include <iterator>
#include <vector>
#include <givaro/modular-floating.h>
#include <givaro/givinteger.h>
#include <givaro/givintprime.h>
#include "givaro/modular-extended.h"
#include <recint/ruint.h>
#include "fflas-ffpack/config-blas.h"
#include "fflas-ffpack/utils/fflas_memory.h"
#include "fflas-ffpack/utils/align-allocator.h"
#include "fflas-ffpack/field/rns-double-elt.h"
#include "rns-double.inl"
#include "rns-double-recint.inl"
```

### Data Structures

- struct [rns\\_double](#)
- struct [rns\\_double\\_extended](#)
- class [rnsRandIter](#)< [RNS](#) >

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*
- namespace [FFLAS](#)

### Macros

- #define [ROUND\\_DOWN](#)(x, s) ((x) & ~((s)-1))

### Functions

- template<> void [fflas\\_delete](#) (FFPACK::rns\_double\_elt\_ptr A)
- template<> void [fflas\\_delete](#) (FFPACK::rns\_double\_elt\_cstptr A)



### 13.212.1 Detailed Description

rns structure with double support

### 13.212.2 Macro Definition Documentation

#### 13.212.2.1 ROUND\_DOWN

```
#define ROUND_DOWN(
 x,
 s) ((x) & ~((s)-1))
```

## 13.213 rns-double.inl File Reference

```
#include "fflas-ffpack/fflas/fflas_freduce.h"
```

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

### Macros

- #define [\\_\\_FFLASFFPACK\\_field\\_rns\\_double\\_INL](#)

### 13.213.1 Macro Definition Documentation

#### 13.213.1.1 \_\_FFLASFFPACK\_field\_rns\_double\_INL

```
#define __FFLASFFPACK_field_rns_double_INL
```

## 13.214 rns-integer-mod.h File Reference

representation of  $\mathbb{Z}/p\mathbb{Z}$  using RNS representation (note: fixed precision)

```
#include <vector>
#include <cmath>
#include <recint/recint.h>
#include <givaro/modular-integer.h>
#include <givaro/givinteger.h>
#include <givaro/udl.h>
#include "givaro/modular-extended.h"
#include "fflas-ffpack/field/rns-double.h"
#include "fflas-ffpack/field/rns-integer.h"
#include "fflas-ffpack/fflas/fflas_level1.inl"
#include "fflas-ffpack/fflas/fflas_level2.inl"
#include "fflas-ffpack/fflas/fflas_level3.inl"
#include "fflas-ffpack/fflas/fflas_enum.h"
#include "fflas-ffpack/fflas/fflas_fscal_mp.inl"
```

### Data Structures

- class [RNSIntegerMod< RNS >](#)
- class [RNSIntegerMod< RNS >::RandIter](#)

## Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*
- namespace [FFLAS](#)

## Functions

- `template<> FFPACK::rns_double_elt_ptr fflas_new` (const [FFPACK::RNSIntegerMod](#)< [FFPACK::rns\\_double](#) > &F, const `size_t` m, const `Alignment` align)
- `template<> FFPACK::rns_double_elt_ptr fflas_new` (const [FFPACK::RNSIntegerMod](#)< [FFPACK::rns\\_double](#) > &F, const `size_t` m, const `size_t` n, const `Alignment` align)
- `template<typename RNS >`  
`void finit_rns` (const [FFPACK::RNSIntegerMod](#)< [RNS](#) > &F, const `size_t` m, const `size_t` n, `size_t` k, const `Givaro::Integer` \*B, const `size_t` ldb, `typename RNS::Element_ptr` A)
- `template<typename RNS >`  
`void finit_trans_rns` (const [FFPACK::RNSIntegerMod](#)< [RNS](#) > &F, const `size_t` m, const `size_t` n, `size_t` k, const `Givaro::Integer` \*B, const `size_t` ldb, `typename RNS::Element_ptr` A)
- `template<typename RNS >`  
`void fconvert_rns` (const [FFPACK::RNSIntegerMod](#)< [RNS](#) > &F, const `size_t` m, const `size_t` n, `Givaro::Integer` alpha, `Givaro::Integer` \*B, const `size_t` ldb, `typename RNS::ConstElement_ptr` A)
- `template<typename RNS >`  
`void fconvert_trans_rns` (const [FFPACK::RNSIntegerMod](#)< [RNS](#) > &F, const `size_t` m, const `size_t` n, `Givaro::Integer` alpha, `Givaro::Integer` \*B, const `size_t` ldb, `typename RNS::ConstElement_ptr` A)

### 13.214.1 Detailed Description

representation of  $\mathbb{Z}/p\mathbb{Z}$  using RNS representation (note: fixed precision)

## 13.215 rns-integer.h File Reference

representation of  $\mathbb{Z}$  using RNS representation (note: fixed precision)

```
#include <givaro/givinteger.h>
#include "fflas-ffpack/field/rns-double.h"
```

## Data Structures

- class [RNSInteger](#)< [RNS](#) >
- class [RNSInteger](#)< [RNS](#) >::RandIter

## Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*
- namespace [FFLAS](#)

## Functions

- `template<> FFPACK::rns_double_elt_ptr fflas_new` (const [FFPACK::RNSInteger](#)< [FFPACK::rns\\_double](#) > &F, const `size_t` m, const `Alignment` align)
- `template<> FFPACK::rns_double_elt_ptr fflas_new` (const [FFPACK::RNSInteger](#)< [FFPACK::rns\\_double](#) > &F, const `size_t` m, const `size_t` n, const `Alignment` align)
- `template<typename RNS >`  
`void finit_rns` (const [FFPACK::RNSInteger](#)< [RNS](#) > &F, const `size_t` m, const `size_t` n, `size_t` k, const `Givaro::Integer` \*B, const `size_t` ldb, `typename FFPACK::RNSInteger< RNS >::Element_ptr A)`
- `template<typename RNS >`  
`void fconvert_rns` (const [FFPACK::RNSInteger](#)< [RNS](#) > &F, const `size_t` m, const `size_t` n, `Givaro::Integer` alpha, `Givaro::Integer` \*B, const `size_t` ldb, `typename FFPACK::RNSInteger< RNS >::ConstElement_ptr A)`

### 13.215.1 Detailed Description

representation of  $\mathbb{Z}$  using RNS representation (note: fixed precision)

## 13.216 rns.h File Reference

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

### 13.217 rns.inl File Reference

```
#include "rns-double.h"
#include "rns-integer.h"
#include "rns-integer-mod.h"
```

### Macros

- #define [\\_\\_FFLASFFPACK\\_field\\_rns\\_INL](#)

### 13.217.1 Macro Definition Documentation

#### 13.217.1.1 \_\_FFLASFFPACK\_field\_rns\_INL

```
#define __FFLASFFPACK_field_rns_INL
```

### 13.218 interfaces.doxy File Reference

### 13.219 fflas\_c.h File Reference

```
#include <stdbool.h>
#include <stdlib.h>
#include <inttypes.h>
```

### Macros

- #define [FFLAS\\_COMPILED](#)

### Enumerations

- enum [FFLAS\\_C\\_ORDER](#) { [FflasRowMajor](#) = 101 , [FflasColMajor](#) = 102 }  
*Storage by row or col ?*
- enum [FFLAS\\_C\\_TRANSPOSE](#) { [FflasNoTrans](#) = 111 , [FflasTrans](#) = 112 }  
*Is matrix transposed ?*
- enum [FFLAS\\_C\\_UPLO](#) { [FflasUpper](#) = 121 , [FflasLower](#) = 122 }  
*Is triangular matrix's shape upper ?*
- enum [FFLAS\\_C\\_DIAG](#) { [FflasNonUnit](#) = 131 , [FflasUnit](#) = 132 }  
*Is the triangular matrix implicitly unit diagonal ?*
- enum [FFLAS\\_C\\_SIDE](#) { [FflasLeft](#) = 141 , [FflasRight](#) = 142 }  
*On what side ?*
- enum [FFLAS\\_C\\_BASE](#) { [FflasDouble](#) = 151 , [FflasFloat](#) = 152 , [FflasGeneric](#) = 153 }  
*FFLAS\_C\_BASE determines the type of the element representation for Matrix Mult kernel.*

## Functions

- void [freducein\\_1\\_modular\\_double](#) (const double p, const size\_t n, double \*X, const size\_t incX, bool positive)
- void [freduce\\_1\\_modular\\_double](#) (const double F, const size\_t n, const double \*Y, const size\_t incY, double \*X, const size\_t incX, bool positive)
- void [fnegin\\_1\\_modular\\_double](#) (const double F, const size\_t n, double \*X, const size\_t incX, bool positive)
- void [fneg\\_1\\_modular\\_double](#) (const double p, const size\_t n, const double \*Y, const size\_t incY, double \*X, const size\_t incX, bool positive)
- void [fzero\\_1\\_modular\\_double](#) (const double p, const size\_t n, double \*X, const size\_t incX, bool positive)
- bool [fiszero\\_1\\_modular\\_double](#) (const double p, const size\_t n, const double \*X, const size\_t incX, bool positive)
- bool [fequal\\_1\\_modular\\_double](#) (const double p, const size\_t n, const double \*X, const size\_t incX, const double \*Y, const size\_t incY, bool positive)
- void [fassign\\_1\\_modular\\_double](#) (const double p, const size\_t n, const double \*Y, const size\_t incY, double \*X, const size\_t incX, bool positive)
- void [fscaln\\_1\\_modular\\_double](#) (const double p, const size\_t n, const double alpha, double \*X, const size\_t incX, bool positive)
- void [fscal\\_1\\_modular\\_double](#) (const double p, const size\_t n, const double alpha, const double \*X, const size\_t incX, double \*Y, const size\_t incY, bool positive)
- void [faxpy\\_1\\_modular\\_double](#) (const double p, const size\_t n, const double alpha, const double \*X, const size\_t incX, double \*Y, const size\_t incY, bool positive)
- double [fdot\\_1\\_modular\\_double](#) (const double p, const size\_t n, const double \*X, const size\_t incX, const double \*Y, const size\_t incY, bool positive)
- void [fswap\\_1\\_modular\\_double](#) (const double p, const size\_t n, double \*X, const size\_t incX, double \*Y, const size\_t incY, bool positive)
- void [fadd\\_1\\_modular\\_double](#) (const double p, const size\_t n, const double \*A, const size\_t incA, const double \*B, const size\_t incB, double \*C, const size\_t incC, bool positive)
- void [fsub\\_1\\_modular\\_double](#) (const double p, const size\_t n, const double \*A, const size\_t incA, const double \*B, const size\_t incB, double \*C, const size\_t incC, bool positive)
- void [faddin\\_1\\_modular\\_double](#) (const double p, const size\_t n, const double \*B, const size\_t incB, double \*C, const size\_t incC, bool positive)
- void [fsubin\\_1\\_modular\\_double](#) (const double p, const size\_t n, const double \*B, const size\_t incB, double \*C, const size\_t incC, bool positive)
- void [fassign\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*B, const size\_t incB, double \*A, const size\_t incA, bool positive)
- void [fzero\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, double \*A, const size\_t incA, bool positive)
- bool [fequal\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*A, const size\_t incA, const double \*B, const size\_t incB, bool positive)
- bool [fiszero\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*A, const size\_t incA, bool positive)
- void [fidentity\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, double \*A, const size\_t incA, const double d, bool positive)
- void [freducein\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, double \*A, const size\_t incA, bool positive)
- void [freduce\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*B, const size\_t incB, double \*A, const size\_t incA, bool positive)
- void [fnegin\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, double \*A, const size\_t incA, bool positive)
- void [fneg\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*B, const size\_t incB, double \*A, const size\_t incA, bool positive)
- void [fscaln\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double alpha, double \*A, const size\_t incA, bool positive)
- void [fscal\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double alpha, const double \*A, const size\_t incA, double \*B, const size\_t incB, bool positive)
- void [faxpy\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double alpha, const double \*X, const size\_t incX, double \*Y, const size\_t incY, bool positive)

- void [fmove\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, double \*A, const size\_t ldA, double \*B, const size\_t ldB, bool positive)
- void [fadd\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*A, const size\_t ldA, const double \*B, const size\_t ldB, double \*C, const size\_t ldC, bool positive)
- void [fsub\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*A, const size\_t ldA, const double \*B, const size\_t ldB, double \*C, const size\_t ldC, bool positive)
- void [fsubin\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*B, const size\_t ldB, double \*C, const size\_t ldC, bool positive)
- void [faddin\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*B, const size\_t ldB, double \*C, const size\_t ldC, bool positive)
- double \* [fgemv\\_2\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_TRANSPOSE](#) TransA, const size\_t m, const size\_t n, const double alpha, const double \*A, const size\_t ldA, const double \*X, const size\_t incX, const double beta, double \*Y, const size\_t incY, bool positive)
- void [fger\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double alpha, const double \*x, const size\_t incX, const double \*y, const size\_t incY, double \*A, const size\_t ldA, bool positive)
- void [ftrsv\\_2\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_UPLO](#) Uplo, const enum [FFLAS\\_C\\_TRANSPOSE](#) TransA, const enum [FFLAS\\_C\\_DIAG](#) Diag, const size\_t n, const double \*A, const size\_t ldA, double \*X, int incX, bool positive)
- void [ftrsm\\_3\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_SIDE](#) Side, const enum [FFLAS\\_C\\_UPLO](#) Uplo, const enum [FFLAS\\_C\\_TRANSPOSE](#) TransA, const enum [FFLAS\\_C\\_DIAG](#) Diag, const size\_t m, const size\_t n, const double alpha, const double \*A, const size\_t ldA, double \*B, const size\_t ldB, bool positive)
- void [ftrmm\\_3\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_SIDE](#) Side, const enum [FFLAS\\_C\\_UPLO](#) Uplo, const enum [FFLAS\\_C\\_TRANSPOSE](#) TransA, const enum [FFLAS\\_C\\_DIAG](#) Diag, const size\_t m, const size\_t n, const double alpha, double \*A, const size\_t ldA, double \*B, const size\_t ldB, bool positive)
- double \* [fgemm\\_3\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_TRANSPOSE](#) tA, const enum [FFLAS\\_C\\_TRANSPOSE](#) tB, const size\_t m, const size\_t n, const size\_t k, const double alpha, const double \*A, const size\_t ldA, const double \*B, const size\_t ldB, const double beta, double \*C, const size\_t ldC, bool positive)
- double \* [fsquare\\_3\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_TRANSPOSE](#) tA, const size\_t n, const double alpha, const double \*A, const size\_t ldA, const double beta, double \*C, const size\_t ldC, bool positive)

## 13.219.1 Macro Definition Documentation

### 13.219.1.1 FFLAS\_COMPILED

```
#define FFLAS_COMPILED
```

## 13.219.2 Enumeration Type Documentation

### 13.219.2.1 FFLAS\_C\_ORDER

```
enum FFLAS_C_ORDER
```

Storage by row or col ?

Enumerator

|                               |           |
|-------------------------------|-----------|
| <a href="#">FflasRowMajor</a> | row major |
| <a href="#">FflasColMajor</a> | col major |

### 13.219.2.2 FFLAS\_C\_TRANSPOSE

```
enum FFLAS_C_TRANSPOSE
```

Is matrix transposed ?

**Enumerator**

|              |                           |
|--------------|---------------------------|
| FflasNoTrans | Matrix is not transposed. |
| FflasTrans   | Matrix is transposed.     |

**13.219.2.3 FFLAS\_C\_UPLO**enum [FFLAS\\_C\\_UPLO](#)

Is triangular matrix's shape upper ?

**Enumerator**

|            |                                                                        |
|------------|------------------------------------------------------------------------|
| FflasUpper | Triangular matrix is Upper triangular (if $i > j$ then $T_{i,j} = 0$ ) |
| FflasLower | Triangular matrix is Lower triangular (if $i < j$ then $T_{i,j} = 0$ ) |

**13.219.2.4 FFLAS\_C\_DIAG**enum [FFLAS\\_C\\_DIAG](#)

Is the triangular matrix implicitly unit diagonal ?

**Enumerator**

|              |                                                                   |
|--------------|-------------------------------------------------------------------|
| FflasNonUnit | Triangular matrix has an explicit arbitrary diagonal.             |
| FflasUnit    | Triangular matrix has an implicit unit diagonal ( $T_{i,i} = 1$ ) |

**13.219.2.5 FFLAS\_C\_SIDE**enum [FFLAS\\_C\\_SIDE](#)

On what side ?

**Enumerator**

|            |                                 |
|------------|---------------------------------|
| FflasLeft  | Operator applied on the left.   |
| FflasRight | Operator applied on the righth. |

**13.219.2.6 FFLAS\_C\_BASE**enum [FFLAS\\_C\\_BASE](#)

FFLAS\_C\_BASE determines the type of the element representation for Matrix Mult kernel.  
(deprecated, should not be used)

**Enumerator**

|              |                                                                            |
|--------------|----------------------------------------------------------------------------|
| FflasDouble  | to use the double precision BLAS                                           |
| FflasFloat   | to use the single precision BLAS                                           |
| FflasGeneric | for any other domain, that can not be converted to floating point integers |

## 13.219.3 Function Documentation

### 13.219.3.1 `freducein_1_modular_double()`

```
void freducein_1_modular_double (
 const double p,
 const size_t n,
 double * X,
 const size_t incX,
 bool positive)
```

### 13.219.3.2 `freduce_1_modular_double()`

```
void freduce_1_modular_double (
 const double F,
 const size_t n,
 const double * Y,
 const size_t incY,
 double * X,
 const size_t incX,
 bool positive)
```

### 13.219.3.3 `fnegin_1_modular_double()`

```
void fnegin_1_modular_double (
 const double F,
 const size_t n,
 double * X,
 const size_t incX,
 bool positive)
```

### 13.219.3.4 `fneg_1_modular_double()`

```
void fneg_1_modular_double (
 const double p,
 const size_t n,
 const double * Y,
 const size_t incY,
 double * X,
 const size_t incX,
 bool positive)
```

### 13.219.3.5 `fzero_1_modular_double()`

```
void fzero_1_modular_double (
 const double p,
 const size_t n,
 double * X,
 const size_t incX,
 bool positive)
```

### 13.219.3.6 `fiszero_1_modular_double()`

```
bool fiszero_1_modular_double (
 const double p,
 const size_t n,
 const double * X,
 const size_t incX,
 bool positive)
```

**13.219.3.7 fequal\_1\_modular\_double()**

```
bool fequal_1_modular_double (
 const double p,
 const size_t n,
 const double * X,
 const size_t incX,
 const double * Y,
 const size_t incY,
 bool positive)
```

**13.219.3.8 fassign\_1\_modular\_double()**

```
void fassign_1_modular_double (
 const double p,
 const size_t n,
 const double * Y,
 const size_t incY,
 double * X,
 const size_t incX,
 bool positive)
```

**13.219.3.9 fscal\_1\_modular\_double()**

```
void fscal_1_modular_double (
 const double p,
 const size_t n,
 const double alpha,
 double * X,
 const size_t incX,
 bool positive)
```

**13.219.3.10 fscale\_1\_modular\_double()**

```
void fscale_1_modular_double (
 const double p,
 const size_t n,
 const double alpha,
 const double * X,
 const size_t incX,
 double * Y,
 const size_t incY,
 bool positive)
```

**13.219.3.11 faxpy\_1\_modular\_double()**

```
void faxpy_1_modular_double (
 const double p,
 const size_t n,
 const double alpha,
 const double * X,
 const size_t incX,
 double * Y,
 const size_t incY,
 bool positive)
```

**13.219.3.12 fdot\_1\_modular\_double()**

```
double fdot_1_modular_double (
 const double p,
```



```
 const size_t n,
 const double * X,
 const size_t incX,
 const double * Y,
 const size_t incY,
 bool positive)
```

#### 13.219.3.13 fswap\_1\_modular\_double()

```
void fswap_1_modular_double (
 const double p,
 const size_t n,
 double * X,
 const size_t incX,
 double * Y,
 const size_t incY,
 bool positive)
```

#### 13.219.3.14 fadd\_1\_modular\_double()

```
void fadd_1_modular_double (
 const double p,
 const size_t n,
 const double * A,
 const size_t incA,
 const double * B,
 const size_t incB,
 double * C,
 const size_t incC,
 bool positive)
```

#### 13.219.3.15 fsub\_1\_modular\_double()

```
void fsub_1_modular_double (
 const double p,
 const size_t n,
 const double * A,
 const size_t incA,
 const double * B,
 const size_t incB,
 double * C,
 const size_t incC,
 bool positive)
```

#### 13.219.3.16 faddin\_1\_modular\_double()

```
void faddin_1_modular_double (
 const double p,
 const size_t n,
 const double * B,
 const size_t incB,
 double * C,
 const size_t incC,
 bool positive)
```

#### 13.219.3.17 fsubin\_1\_modular\_double()

```
void fsubin_1_modular_double (
 const double p,
```

```
 const size_t n,
 const double * B,
 const size_t incB,
 double * C,
 const size_t incC,
 bool positive)
```

#### 13.219.3.18 fassign\_2\_modular\_double()

```
void fassign_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 const double * B,
 const size_t ldB,
 double * A,
 const size_t ldA,
 bool positive)
```

#### 13.219.3.19 fzero\_2\_modular\_double()

```
void fzero_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 double * A,
 const size_t ldA,
 bool positive)
```

#### 13.219.3.20 fequal\_2\_modular\_double()

```
bool fequal_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 const double * A,
 const size_t ldA,
 const double * B,
 const size_t ldB,
 bool positive)
```

#### 13.219.3.21 fiszero\_2\_modular\_double()

```
bool fiszero_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 const double * A,
 const size_t ldA,
 bool positive)
```

#### 13.219.3.22 fidentity\_2\_modular\_double()

```
void fidentity_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 double * A,
 const size_t ldA,
```

```
 const double d,
 bool positive)
```

#### 13.219.3.23 freducein\_2\_modular\_double()

```
void freducein_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 double * A,
 const size_t ldA,
 bool positive)
```

#### 13.219.3.24 freduce\_2\_modular\_double()

```
void freduce_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 const double * B,
 const size_t ldB,
 double * A,
 const size_t ldA,
 bool positive)
```

#### 13.219.3.25 fnegin\_2\_modular\_double()

```
void fnegin_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 double * A,
 const size_t ldA,
 bool positive)
```

#### 13.219.3.26 fneg\_2\_modular\_double()

```
void fneg_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 const double * B,
 const size_t ldB,
 double * A,
 const size_t ldA,
 bool positive)
```

#### 13.219.3.27 fscaln\_2\_modular\_double()

```
void fscaln_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 const double alpha,
 double * A,
 const size_t ldA,
 bool positive)
```

**13.219.3.28 fscal\_2\_modular\_double()**

```
void fscal_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 const double alpha,
 const double * A,
 const size_t ldA,
 double * B,
 const size_t ldB,
 bool positive)
```

**13.219.3.29 faxpy\_2\_modular\_double()**

```
void faxpy_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 const double alpha,
 const double * X,
 const size_t ldX,
 double * Y,
 const size_t ldY,
 bool positive)
```

**13.219.3.30 fmove\_2\_modular\_double()**

```
void fmove_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 double * A,
 const size_t ldA,
 double * B,
 const size_t ldB,
 bool positive)
```

**13.219.3.31 fadd\_2\_modular\_double()**

```
void fadd_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 const double * A,
 const size_t ldA,
 const double * B,
 const size_t ldB,
 double * C,
 const size_t ldC,
 bool positive)
```

**13.219.3.32 fsub\_2\_modular\_double()**

```
void fsub_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 const double * A,
```

```

 const size_t ldA,
 const double * B,
 const size_t ldB,
 double * C,
 const size_t ldC,
 bool positive)

```

### 13.219.3.33 fsubin\_2\_modular\_double()

```

void fsubin_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 const double * B,
 const size_t ldB,
 double * C,
 const size_t ldC,
 bool positive)

```

### 13.219.3.34 faddin\_2\_modular\_double()

```

void faddin_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 const double * B,
 const size_t ldB,
 double * C,
 const size_t ldC,
 bool positive)

```

### 13.219.3.35 fgemv\_2\_modular\_double()

```

double * fgemv_2_modular_double (
 const double p,
 const enum FFLAS_C_TRANSPOSE TransA,
 const size_t m,
 const size_t n,
 const double alpha,
 const double * A,
 const size_t ldA,
 const double * X,
 const size_t incX,
 const double betA,
 double * Y,
 const size_t incY,
 bool positive)

```

### 13.219.3.36 fger\_2\_modular\_double()

```

void fger_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 const double alpha,
 const double * x,
 const size_t incX,
 const double * y,
 const size_t incY,

```

```
double * A,
const size_t ldA,
bool positive)
```

### 13.219.3.37 ftrsv\_2\_modular\_double()

```
void ftrsv_2_modular_double (
 const double p,
 const enum FFLAS_C_UPLO Uplo,
 const enum FFLAS_C_TRANSPOSE TransA,
 const enum FFLAS_C_DIAG Diag,
 const size_t n,
 const double * A,
 const size_t ldA,
 double * X,
 int incX,
 bool positive)
```

### 13.219.3.38 ftrsm\_3\_modular\_double()

```
void ftrsm_3_modular_double (
 const double p,
 const enum FFLAS_C_SIDE Side,
 const enum FFLAS_C_UPLO Uplo,
 const enum FFLAS_C_TRANSPOSE TransA,
 const enum FFLAS_C_DIAG Diag,
 const size_t m,
 const size_t n,
 const double alpha,
 const double * A,
 const size_t ldA,
 double * B,
 const size_t ldB,
 bool positive)
```

### 13.219.3.39 ftrmm\_3\_modular\_double()

```
void ftrmm_3_modular_double (
 const double p,
 const enum FFLAS_C_SIDE Side,
 const enum FFLAS_C_UPLO Uplo,
 const enum FFLAS_C_TRANSPOSE TransA,
 const enum FFLAS_C_DIAG Diag,
 const size_t m,
 const size_t n,
 const double alpha,
 double * A,
 const size_t ldA,
 double * B,
 const size_t ldB,
 bool positive)
```

### 13.219.3.40 fgemm\_3\_modular\_double()

```
double * fgemm_3_modular_double (
 const double p,
 const enum FFLAS_C_TRANSPOSE tA,
 const enum FFLAS_C_TRANSPOSE tB,
 const size_t m,
```

```

 const size_t n,
 const size_t k,
 const double alpha,
 const double * A,
 const size_t ldA,
 const double * B,
 const size_t ldB,
 const double betA,
 double * C,
 const size_t ldC,
 bool positive)

```

#### 13.219.3.41 fsquare\_3\_modular\_double()

```

double * fsquare_3_modular_double (
 const double p,
 const enum FFLAS_C_TRANSPOSE tA,
 const size_t n,
 const double alpha,
 const double * A,
 const size_t ldA,
 const double betA,
 double * C,
 const size_t ldC,
 bool positive)

```

## 13.220 fflas\_L1\_inst.C File Reference

```

#include "fflas-ffpack/fflas-ffpack-config.h"
#include "givaro/modular.h"
#include "givaro/modular-balanced.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/fflas/fflas_helpers.inl"
#include "fflas_L1_inst_implem.inl"

```

### Macros

- `#define __FFLAS_L1_INST_C`
- `#define INST_OR_DECL`
- `#define FFLAS_FIELD Givaro::ModularBalanced`
- `#define FFLAS_ELT double`
- `#define FFLAS_ELT float`
- `#define FFLAS_ELT int64_t`
- `#define FFLAS_FIELD Givaro::Modular`
- `#define FFLAS_ELT double`
- `#define FFLAS_ELT float`
- `#define FFLAS_ELT int64_t`

### 13.220.1 Macro Definition Documentation

#### 13.220.1.1 \_\_FFLAS\_L1\_INST\_C

```
#define __FFLAS_L1_INST_C
```

#### 13.220.1.2 INST\_OR\_DECL

```
#define INST_OR_DECL
```

**13.220.1.3 FFLAS\_FIELD [1/2]**

```
#define FFLAS_FIELD Givaro::ModularBalanced
```

**13.220.1.4 FFLAS\_ELT [1/6]**

```
#define FFLAS_ELT double
```

**13.220.1.5 FFLAS\_ELT [2/6]**

```
#define FFLAS_ELT float
```

**13.220.1.6 FFLAS\_ELT [3/6]**

```
#define FFLAS_ELT int64_t
```

**13.220.1.7 FFLAS\_FIELD [2/2]**

```
#define FFLAS_FIELD Givaro::Modular
```

**13.220.1.8 FFLAS\_ELT [4/6]**

```
#define FFLAS_ELT double
```

**13.220.1.9 FFLAS\_ELT [5/6]**

```
#define FFLAS_ELT float
```

**13.220.1.10 FFLAS\_ELT [6/6]**

```
#define FFLAS_ELT int64_t
```

**13.221 fflas\_L1\_inst.h File Reference**

```
#include "givaro/modular.h"
#include "givaro/modular-balanced.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/fflas/fflas_helpers.inl"
#include "fflas_L1_inst_implem.inl"
```

**Macros**

- #define [INST\\_OR\\_DECL](#) <>
- #define [FFLAS\\_FIELD](#) Givaro::ModularBalanced
- #define [FFLAS\\_ELT](#) double
- #define [FFLAS\\_ELT](#) float
- #define [FFLAS\\_ELT](#) int64\_t
- #define [FFLAS\\_FIELD](#) Givaro::Modular
- #define [FFLAS\\_ELT](#) double
- #define [FFLAS\\_ELT](#) float
- #define [FFLAS\\_ELT](#) int64\_t

**13.221.1 Macro Definition Documentation****13.221.1.1 INST\_OR\_DECL**

```
#define INST_OR_DECL <>
```



**13.221.1.2 FFLAS\_FIELD [1/2]**

```
#define FFLAS_FIELD Givaro::ModularBalanced
```

**13.221.1.3 FFLAS\_ELT [1/6]**

```
#define FFLAS_ELT double
```

**13.221.1.4 FFLAS\_ELT [2/6]**

```
#define FFLAS_ELT float
```

**13.221.1.5 FFLAS\_ELT [3/6]**

```
#define FFLAS_ELT int64_t
```

**13.221.1.6 FFLAS\_FIELD [2/2]**

```
#define FFLAS_FIELD Givaro::Modular
```

**13.221.1.7 FFLAS\_ELT [4/6]**

```
#define FFLAS_ELT double
```

**13.221.1.8 FFLAS\_ELT [5/6]**

```
#define FFLAS_ELT float
```

**13.221.1.9 FFLAS\_ELT [6/6]**

```
#define FFLAS_ELT int64_t
```

**13.222 fflas\_L1\_inst\_implem.inl File Reference****Namespaces**

- namespace [FFLAS](#)

**Functions**

- template [INST\\_OR\\_DECL](#) void [freduce](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const size\_t n, [FFLAS\\_ELT](#) \*X, const size\_t incX)  

$$\text{freduce } x \leftarrow x \bmod F.$$
- template [INST\\_OR\\_DECL](#) void [freduce](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const size\_t n, const [FFLAS\\_ELT](#) \*Y, const size\_t incY, [FFLAS\\_ELT](#) \*X, const size\_t incX)  

$$\text{freduce } x \leftarrow y \bmod F.$$
- template [INST\\_OR\\_DECL](#) void [finit](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const size\_t n, const [FFLAS\\_ELT](#) \*Y, const size\_t incY, [FFLAS\\_ELT](#) \*X, const size\_t incX)  

$$\text{finit } x \leftarrow y \bmod F.$$
- template [INST\\_OR\\_DECL](#) void [fconvert](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const size\_t n, [FFLAS\\_ELT](#) \*X, const size\_t incX, const [FFLAS\\_ELT](#) \*Y, const size\_t incY)  

$$\text{fconvert } x \leftarrow y \bmod F.$$
- template [INST\\_OR\\_DECL](#) void [fnegin](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const size\_t n, [FFLAS\\_ELT](#) \*X, const size\_t incX)  

$$\text{fnegin } x \leftarrow -x.$$
- template [INST\\_OR\\_DECL](#) void [fneg](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const size\_t n, const [FFLAS\\_ELT](#) \*Y, const size\_t incY, [FFLAS\\_ELT](#) \*X, const size\_t incX)  

$$\text{fneg } x \leftarrow -y.$$

- template `INST_OR_DECL` void `fzero` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t n, `FFLAS_ELT` \*X, const size\_t incX)  
 $fzero : A \leftarrow 0.$
- template `INST_OR_DECL` bool `fiszero` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t n, const `FFLAS_ELT` \*X, const size\_t incX)  
 $fiszero : test\ X = 0.$
- template `INST_OR_DECL` bool `fequal` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t n, const `FFLAS_ELT` \*X, const size\_t incX, const `FFLAS_ELT` \*Y, const size\_t incY)  
 $fequal : test\ X = Y.$
- template `INST_OR_DECL` void `fassign` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t N, const `FFLAS_ELT` \*Y, const size\_t incY, `FFLAS_ELT` \*X, const size\_t incX)  
 $fassign : x \leftarrow y.$
- template `INST_OR_DECL` void `fscaln` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t n, const `FFLAS_ELT` alpha, `FFLAS_ELT` \*X, const size\_t incX)  
 $fscaln\ x \leftarrow \alpha \cdot x.$
- template `INST_OR_DECL` void `fscal` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t n, const `FFLAS_ELT` alpha, const `FFLAS_ELT` \*X, const size\_t incX, `FFLAS_ELT` \*Y, const size\_t incY)  
 $fscal\ y \leftarrow \alpha \cdot x.$
- template `INST_OR_DECL` void `faxpy` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t N, const `FFLAS_ELT` alpha, const `FFLAS_ELT` \*X, const size\_t incX, `FFLAS_ELT` \*Y, const size\_t incY)  
 $faxpy : y \leftarrow \alpha \cdot x + y.$
- template `INST_OR_DECL` `FFLAS_ELT` `fdot` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t N, const `FFLAS_ELT` \*X, const size\_t incX, const `FFLAS_ELT` \*Y, const size\_t incY)  
 $faxpy : y \leftarrow \alpha \cdot x + \beta \cdot y.$
- template `INST_OR_DECL` void `fswap` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t N, `FFLAS_ELT` \*X, const size\_t incX, `FFLAS_ELT` \*Y, const size\_t incY)  
 $fswap : X \leftrightarrow Y.$
- template `INST_OR_DECL` void `fadd` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t N, const `FFLAS_ELT` \*A, const size\_t inca, const `FFLAS_ELT` \*B, const size\_t incb, `FFLAS_ELT` \*C, const size\_t incc)
- template `INST_OR_DECL` void `fsub` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t N, const `FFLAS_ELT` \*A, const size\_t inca, const `FFLAS_ELT` \*B, const size\_t incb, `FFLAS_ELT` \*C, const size\_t incc)
- template `INST_OR_DECL` void `faddin` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t N, const `FFLAS_ELT` \*B, const size\_t incb, `FFLAS_ELT` \*C, const size\_t incc)
- template `INST_OR_DECL` void `fadd` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t N, const `FFLAS_ELT` \*A, const size\_t inca, const `FFLAS_ELT` alpha, const `FFLAS_ELT` \*B, const size\_t incb, `FFLAS_ELT` \*C, const size\_t incc)

### 13.223 fflas\_L2\_inst.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "givaro/modular.h"
#include "givaro/modular-balanced.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/fflas/fflas_helpers.inl"
#include "fflas_L2_inst_implem.inl"
```

#### Macros

- #define `__FFLAS_L2_INST_C`
- #define `INST_OR_DECL`
- #define `FFLAS_FIELD` Givaro::ModularBalanced

- `#define FFLAS_ELT` double
- `#define FFLAS_ELT` float
- `#define FFLAS_ELT` int64\_t
- `#define FFLAS_FIELD` Givaro::Modular
- `#define FFLAS_ELT` double
- `#define FFLAS_ELT` float
- `#define FFLAS_ELT` int64\_t

### 13.223.1 Macro Definition Documentation

#### 13.223.1.1 \_\_FFLAS\_L2\_INST\_C

```
#define __FFLAS_L2_INST_C
```

#### 13.223.1.2 INST\_OR\_DECL

```
#define INST_OR_DECL
```

#### 13.223.1.3 FFLAS\_FIELD [1/2]

```
#define FFLAS_FIELD Givaro::ModularBalanced
```

#### 13.223.1.4 FFLAS\_ELT [1/6]

```
#define FFLAS_ELT double
```

#### 13.223.1.5 FFLAS\_ELT [2/6]

```
#define FFLAS_ELT float
```

#### 13.223.1.6 FFLAS\_ELT [3/6]

```
#define FFLAS_ELT int64_t
```

#### 13.223.1.7 FFLAS\_FIELD [2/2]

```
#define FFLAS_FIELD Givaro::Modular
```

#### 13.223.1.8 FFLAS\_ELT [4/6]

```
#define FFLAS_ELT double
```

#### 13.223.1.9 FFLAS\_ELT [5/6]

```
#define FFLAS_ELT float
```

#### 13.223.1.10 FFLAS\_ELT [6/6]

```
#define FFLAS_ELT int64_t
```

## 13.224 fflas\_L2\_inst.h File Reference

```
#include "givaro/modular.h"
#include "givaro/modular-balanced.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/fflas/fflas_helpers.inl"
#include "fflas_L2_inst_implem.inl"
```

## Macros

- `#define INST_OR_DECL <>`
- `#define FFLAS_FIELD Givaro::ModularBalanced`
- `#define FFLAS_ELT double`
- `#define FFLAS_ELT float`
- `#define FFLAS_ELT int64_t`
- `#define FFLAS_FIELD Givaro::Modular`
- `#define FFLAS_ELT double`
- `#define FFLAS_ELT float`
- `#define FFLAS_ELT int64_t`

## 13.224.1 Macro Definition Documentation

### 13.224.1.1 INST\_OR\_DECL

```
#define INST_OR_DECL <>
```

### 13.224.1.2 FFLAS\_FIELD [1/2]

```
#define FFLAS_FIELD Givaro::ModularBalanced
```

### 13.224.1.3 FFLAS\_ELT [1/6]

```
#define FFLAS_ELT double
```

### 13.224.1.4 FFLAS\_ELT [2/6]

```
#define FFLAS_ELT float
```

### 13.224.1.5 FFLAS\_ELT [3/6]

```
#define FFLAS_ELT int64_t
```

### 13.224.1.6 FFLAS\_FIELD [2/2]

```
#define FFLAS_FIELD Givaro::Modular
```

### 13.224.1.7 FFLAS\_ELT [4/6]

```
#define FFLAS_ELT double
```

### 13.224.1.8 FFLAS\_ELT [5/6]

```
#define FFLAS_ELT float
```

### 13.224.1.9 FFLAS\_ELT [6/6]

```
#define FFLAS_ELT int64_t
```

## 13.225 fflas\_L2\_inst\_implem.inl File Reference

### Namespaces

- namespace `FFLAS`

## Functions

- template `INST_OR_DECL` void `fassign` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, const `FFLAS_ELT` \*B, const size\_t ldb, `FFLAS_ELT` \*A, const size\_t lda)  
 $fassign : A \leftarrow B.$
- template `INST_OR_DECL` void `fzero` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, `FFLAS_ELT` \*A, const size\_t lda)  
 $fzero : A \leftarrow 0.$
- template `INST_OR_DECL` bool `fequal` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, const `FFLAS_ELT` \*A, const size\_t lda, const `FFLAS_ELT` \*B, const size\_t ldb)  
 $fequal : test A = B.$
- template `INST_OR_DECL` bool `fiszero` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, const `FFLAS_ELT` \*A, const size\_t lda)  
 $fiszero : test A = 0.$
- template `INST_OR_DECL` void `fidentity` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, `FFLAS_ELT` \*A, const size\_t lda, const `FFLAS_ELT` &d)  
 $creates\ a\ diagonal\ matrix$
- template `INST_OR_DECL` void `fidentity` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, `FFLAS_ELT` \*A, const size\_t lda)  
 $creates\ a\ diagonal\ matrix$
- template `INST_OR_DECL` void `freduce` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, `FFLAS_ELT` \*A, const size\_t lda)  
 $freduce\ A \leftarrow A mod F.$
- template `INST_OR_DECL` void `freduce` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, const `FFLAS_ELT` \*B, const size\_t ldb, `FFLAS_ELT` \*A, const size\_t lda)  
 $freduce\ A \leftarrow B mod F.$
- template `INST_OR_DECL` void `finit` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, const `FFLAS_ELT` \*B, const size\_t ldb, `FFLAS_ELT` \*A, const size\_t lda)  
 $finit\ A \leftarrow B mod F.$
- template `INST_OR_DECL` void `fnegin` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, `FFLAS_ELT` \*A, const size\_t lda)  
 $fnegin\ A \leftarrow -A.$
- template `INST_OR_DECL` void `fneg` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, const `FFLAS_ELT` \*B, const size\_t ldb, `FFLAS_ELT` \*A, const size\_t lda)  
 $fneg\ A \leftarrow -B.$
- template `INST_OR_DECL` void `fscal` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, const `FFLAS_ELT` alpha, `FFLAS_ELT` \*A, const size\_t lda)  
 $fscal\ A \leftarrow a \cdot A.$
- template `INST_OR_DECL` void `fscal` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, const `FFLAS_ELT` alpha, const `FFLAS_ELT` \*A, const size\_t lda, `FFLAS_ELT` \*B, const size\_t ldb)  
 $fscal\ B \leftarrow a \cdot A.$
- template `INST_OR_DECL` void `faxpy` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, const `FFLAS_ELT` alpha, const `FFLAS_ELT` \*X, const size\_t ldx, `FFLAS_ELT` \*Y, const size\_t ldy)  
 $faxpy : y \leftarrow \alpha \cdot x + y.$
- template `INST_OR_DECL` void `fmove` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t m, const size\_t n, `FFLAS_ELT` \*A, const size\_t lda, `FFLAS_ELT` \*B, const size\_t ldb)  
 $faxpy : y \leftarrow \alpha \cdot x + \beta \cdot y.$
- template `INST_OR_DECL` void `fadd` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t M, const size\_t N, const `FFLAS_ELT` \*A, const size\_t lda, const `FFLAS_ELT` \*B, const size\_t ldb, `FFLAS_ELT` \*C, const size\_t ldc)  
 $fadd : matrix\ addition.$
- template `INST_OR_DECL` void `fsub` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t M, const size\_t N, const `FFLAS_ELT` \*A, const size\_t lda, const `FFLAS_ELT` \*B, const size\_t ldb, `FFLAS_ELT` \*C, const size\_t ldc)

- fsub* : matrix subtraction.
- template `INST_OR_DECL` void `fsubin` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t M, const size\_t N, const `FFLAS_ELT` \*B, const size\_t ldb, `FFLAS_ELT` \*C, const size\_t ldc)
- fsubin*  $C = C - B$
- template `INST_OR_DECL` void `fadd` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t M, const size\_t N, const `FFLAS_ELT` \*A, const size\_t lda, const `FFLAS_ELT` alpha, const `FFLAS_ELT` \*B, const size\_t ldb, `FFLAS_ELT` \*C, const size\_t ldc)
- fadd* : matrix addition with scaling.
- template `INST_OR_DECL` void `faddin` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t M, const size\_t N, const `FFLAS_ELT` \*B, const size\_t ldb, `FFLAS_ELT` \*C, const size\_t ldc)
- faddin*
- template `INST_OR_DECL` `FFLAS_ELT` \* `fgemv` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `FFLAS_TRANSPOSE` TransA, const size\_t M, const size\_t N, const `FFLAS_ELT` alpha, const `FFLAS_ELT` \*A, const size\_t lda, const `FFLAS_ELT` \*X, const size\_t incX, const `FFLAS_ELT` beta, `FFLAS_ELT` \*Y, const size\_t incY)
- finite prime FFLAS\_FIELD*<*FFLAS\_ELT*> *GEneral Matrix Vector multiplication.*
- template `INST_OR_DECL` void `fger` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t M, const size\_t N, const `FFLAS_ELT` alpha, const `FFLAS_ELT` \*x, const size\_t incx, const `FFLAS_ELT` \*y, const size\_t incy, `FFLAS_ELT` \*A, const size\_t lda)
- fger*: rank one update of a general matrix
- template `INST_OR_DECL` void `ftsv` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `FFLAS_UPLO` Uplo, const `FFLAS_TRANSPOSE` TransA, const `FFLAS_DIAG` Diag, const size\_t N, const `FFLAS_ELT` \*A, const size\_t lda, `FFLAS_ELT` \*X, int incX)
- ftsv*: *TRI*angular System solve with Vector Computes  $X \leftarrow \text{op}(A^{-1})X$

## 13.226 fflas\_L3\_inst.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "givaro/modular.h"
#include "givaro/modular-balanced.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/fflas/fflas_helpers.inl"
#include "fflas_L3_inst_implem.inl"
```

### Macros

- #define `__FFLAS_L3_INST_C`
- #define `INST_OR_DECL`
- #define `FFLAS_FIELD` Givaro::ModularBalanced
- #define `FFLAS_ELT` double
- #define `FFLAS_ELT` float
- #define `FFLAS_ELT` int64\_t
- #define `FFLAS_FIELD` Givaro::Modular
- #define `FFLAS_ELT` double
- #define `FFLAS_ELT` float
- #define `FFLAS_ELT` int64\_t

### 13.226.1 Macro Definition Documentation

#### 13.226.1.1 \_\_FFLAS\_L3\_INST\_C

```
#define __FFLAS_L3_INST_C
```

**13.226.1.2 INST\_OR\_DECL**

```
#define INST_OR_DECL
```

**13.226.1.3 FFLAS\_FIELD [1/2]**

```
#define FFLAS_FIELD Givaro::ModularBalanced
```

**13.226.1.4 FFLAS\_ELT [1/6]**

```
#define FFLAS_ELT double
```

**13.226.1.5 FFLAS\_ELT [2/6]**

```
#define FFLAS_ELT float
```

**13.226.1.6 FFLAS\_ELT [3/6]**

```
#define FFLAS_ELT int64_t
```

**13.226.1.7 FFLAS\_FIELD [2/2]**

```
#define FFLAS_FIELD Givaro::Modular
```

**13.226.1.8 FFLAS\_ELT [4/6]**

```
#define FFLAS_ELT double
```

**13.226.1.9 FFLAS\_ELT [5/6]**

```
#define FFLAS_ELT float
```

**13.226.1.10 FFLAS\_ELT [6/6]**

```
#define FFLAS_ELT int64_t
```

**13.227 fflas\_L3\_inst.h File Reference**

```
#include "givaro/modular.h"
#include "givaro/modular-balanced.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/fflas/fflas_helpers.inl"
#include "fflas_L3_inst_implem.inl"
```

**Macros**

- `#define INST_OR_DECL <>`
- `#define FFLAS_FIELD Givaro::ModularBalanced`
- `#define FFLAS_ELT double`
- `#define FFLAS_ELT float`
- `#define FFLAS_ELT int64_t`
- `#define FFLAS_FIELD Givaro::Modular`
- `#define FFLAS_ELT double`
- `#define FFLAS_ELT float`
- `#define FFLAS_ELT int64_t`

### 13.227.1 Macro Definition Documentation

#### 13.227.1.1 INST\_OR\_DECL

```
#define INST_OR_DECL <>
```

#### 13.227.1.2 FFLAS\_FIELD [1/2]

```
#define FFLAS_FIELD Givaro::ModularBalanced
```

#### 13.227.1.3 FFLAS\_ELT [1/6]

```
#define FFLAS_ELT double
```

#### 13.227.1.4 FFLAS\_ELT [2/6]

```
#define FFLAS_ELT float
```

#### 13.227.1.5 FFLAS\_ELT [3/6]

```
#define FFLAS_ELT int64_t
```

#### 13.227.1.6 FFLAS\_FIELD [2/2]

```
#define FFLAS_FIELD Givaro::Modular
```

#### 13.227.1.7 FFLAS\_ELT [4/6]

```
#define FFLAS_ELT double
```

#### 13.227.1.8 FFLAS\_ELT [5/6]

```
#define FFLAS_ELT float
```

#### 13.227.1.9 FFLAS\_ELT [6/6]

```
#define FFLAS_ELT int64_t
```

## 13.228 fflas\_L3\_inst\_implem.inl File Reference

### Namespaces

- namespace [FFLAS](#)

### Macros

- #define [\\_\\_FFLAS\\_TRSM\\_READONLY](#)

### Functions

- template [INST\\_OR\\_DECL](#) void [ftrsm](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const [FFLAS\\_SIDE](#) Side, const [FFLAS\\_UPLO](#) Uplo, const [FFLAS\\_TRANSPOSE](#) TransA, const [FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, const [FFLAS\\_ELT](#) alpha, const [FFLAS\\_ELT](#) \*A, const size\_t lda, [FFLAS\\_ELT](#) \*B, const size\_t ldb)  
*ftrsm: **TRI**angular **S**ystem solve with **M**atrix.*
- template [INST\\_OR\\_DECL](#) void [ftrmm](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const [FFLAS\\_SIDE](#) Side, const [FFLAS\\_UPLO](#) Uplo, const [FFLAS\\_TRANSPOSE](#) TransA, const [FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, const [FFLAS\\_ELT](#) alpha, const [FFLAS\\_ELT](#) \*A, const size\_t lda, [FFLAS\\_ELT](#) \*B, const size\_t ldb)  
*ftrmm: **TRI**angular **M**atrix **M**ultiply.*



- template `INST_OR_DECL FFLAS_ELT * fgemm (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const FFLAS_ELT alpha, const FFLAS_ELT *A, const size_t lda, const FFLAS_ELT *B, const size_t ldb, const FFLAS_ELT beta, FFLAS_ELT *C, const size_t ldc)`

*fgemm: Field **GE**neral **M**atrix **M**ultiply.*

- template `INST_OR_DECL FFLAS_ELT * fgemm (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const FFLAS_ELT alpha, const FFLAS_ELT *A, const size_t lda, const FFLAS_ELT *B, const size_t ldb, const FFLAS_ELT beta, FFLAS_ELT *C, const size_t ldc, const ParSeqHelper::Sequential seq)`
- template `INST_OR_DECL FFLAS_ELT * fgemm (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const FFLAS_ELT alpha, const FFLAS_ELT *A, const size_t lda, const FFLAS_ELT *B, const size_t ldb, const FFLAS_ELT beta, FFLAS_ELT *C, const size_t ldc, const ParSeqHelper::Parallel< CuttingStrategy::Recursive, StrategyParameter::TwoDAdaptive > par)`
- template `INST_OR_DECL FFLAS_ELT * fgemm (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb, const size_t m, const size_t n, const size_t k, const FFLAS_ELT alpha, const FFLAS_ELT *A, const size_t lda, const FFLAS_ELT *B, const size_t ldb, const FFLAS_ELT beta, FFLAS_ELT *C, const size_t ldc, const ParSeqHelper::Parallel< CuttingStrategy::Block, StrategyParameter::Threads > par)`
- template `INST_OR_DECL FFLAS_ELT * fsquare (const FFLAS_FIELD< FFLAS_ELT > &F, const FFLAS_TRANSPOSE ta, const size_t n, const FFLAS_ELT alpha, const FFLAS_ELT *A, const size_t lda, const FFLAS_ELT beta, FFLAS_ELT *C, const size_t ldc)`

*fsquare: Squares a matrix.*

## 13.228.1 Macro Definition Documentation

### 13.228.1.1 \_\_FFLAS\_TRSM\_READONLY

```
#define __FFLAS_TRSM_READONLY
```

## 13.229 fflas\_lvl1.C File Reference

C functions calls for level 1 `FFLAS` in `fflas-c.h`.

```
#include "fflas-ffpack/interfaces/libs/fflas_c.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "givaro/modular-balanced.h"
#include "givaro/modular.h"
```

### Functions

- void `freducein_1_modular_double` (const double p, const size\_t n, double \*X, const size\_t incX, bool positive)
- void `freduce_1_modular_double` (const double p, const size\_t n, const double \*Y, const size\_t incY, double \*X, const size\_t incX, bool positive)
- void `fnegin_1_modular_double` (const double p, const size\_t n, double \*X, const size\_t incX, bool positive)
- void `fneg_1_modular_double` (const double p, const size\_t n, const double \*Y, const size\_t incY, double \*X, const size\_t incX, bool positive)
- void `fzero_1_modular_double` (const double p, const size\_t n, double \*X, const size\_t incX, bool positive)
- bool `fiszero_1_modular_double` (const double p, const size\_t n, const double \*X, const size\_t incX, bool positive)
- bool `fequal_1_modular_double` (const double p, const size\_t n, const double \*X, const size\_t incX, const double \*Y, const size\_t incY, bool positive)
- void `fassign_1_modular_double` (const double p, const size\_t n, const double \*Y, const size\_t incY, double \*X, const size\_t incX, bool positive)
- void `fscaln_1_modular_double` (const double p, const size\_t n, const double alpha, double \*X, const size\_t incX, bool positive)

- void [fscal\\_1\\_modular\\_double](#) (const double p, const size\_t n, const double alpha, const double \*X, const size\_t incX, double \*Y, const size\_t incY, bool positive)
- void [faxpy\\_1\\_modular\\_double](#) (const double p, const size\_t n, const double alpha, const double \*X, const size\_t incX, double \*Y, const size\_t incY, bool positive)
- double [fdot\\_1\\_modular\\_double](#) (const double p, const size\_t n, const double \*X, const size\_t incX, const double \*Y, const size\_t incY, bool positive)
- void [fswap\\_1\\_modular\\_double](#) (const double p, const size\_t n, double \*X, const size\_t incX, double \*Y, const size\_t incY, bool positive)
- void [fadd\\_1\\_modular\\_double](#) (const double p, const size\_t n, const double \*A, const size\_t incA, const double \*B, const size\_t incB, double \*C, const size\_t incC, bool positive)
- void [fsub\\_1\\_modular\\_double](#) (const double p, const size\_t n, const double \*A, const size\_t incA, const double \*B, const size\_t incB, double \*C, const size\_t incC, bool positive)
- void [faddin\\_1\\_modular\\_double](#) (const double p, const size\_t n, const double \*B, const size\_t incB, double \*C, const size\_t incC, bool positive)
- void [fsubin\\_1\\_modular\\_double](#) (const double p, const size\_t n, const double \*B, const size\_t incB, double \*C, const size\_t incC, bool positive)

### 13.229.1 Detailed Description

C functions calls for level 1 [FFLAS](#) in `fflas-c.h`.

Author

Brice Boyer

See also

[fflas/fflas\\_level1.inl](#)

### 13.229.2 Function Documentation

#### 13.229.2.1 `freducein_1_modular_double()`

```
void freducein_1_modular_double (
 const double p,
 const size_t n,
 double * X,
 const size_t incX,
 bool positive)
```

#### 13.229.2.2 `freduce_1_modular_double()`

```
void freduce_1_modular_double (
 const double p,
 const size_t n,
 const double * Y,
 const size_t incY,
 double * X,
 const size_t incX,
 bool positive)
```

#### 13.229.2.3 `fnegin_1_modular_double()`

```
void fnegin_1_modular_double (
 const double p,
 const size_t n,
 double * X,
 const size_t incX,
 bool positive)
```

**13.229.2.4 fneg\_1\_modular\_double()**

```
void fneg_1_modular_double (
 const double p,
 const size_t n,
 const double * Y,
 const size_t incY,
 double * X,
 const size_t incX,
 bool positive)
```

**13.229.2.5 fzero\_1\_modular\_double()**

```
void fzero_1_modular_double (
 const double p,
 const size_t n,
 double * X,
 const size_t incX,
 bool positive)
```

**13.229.2.6 fiszero\_1\_modular\_double()**

```
bool fiszero_1_modular_double (
 const double p,
 const size_t n,
 const double * X,
 const size_t incX,
 bool positive)
```

**13.229.2.7 fequal\_1\_modular\_double()**

```
bool fequal_1_modular_double (
 const double p,
 const size_t n,
 const double * X,
 const size_t incX,
 const double * Y,
 const size_t incY,
 bool positive)
```

**13.229.2.8 fassign\_1\_modular\_double()**

```
void fassign_1_modular_double (
 const double p,
 const size_t n,
 const double * Y,
 const size_t incY,
 double * X,
 const size_t incX,
 bool positive)
```

**13.229.2.9 fscaln\_1\_modular\_double()**

```
void fscaln_1_modular_double (
 const double p,
 const size_t n,
 const double alpha,
 double * X,
 const size_t incX,
 bool positive)
```

**13.229.2.10 fscal\_1\_modular\_double()**

```
void fscal_1_modular_double (
 const double p,
 const size_t n,
 const double alpha,
 const double * X,
 const size_t incX,
 double * Y,
 const size_t incY,
 bool positive)
```

**13.229.2.11 faxpy\_1\_modular\_double()**

```
void faxpy_1_modular_double (
 const double p,
 const size_t n,
 const double alpha,
 const double * X,
 const size_t incX,
 double * Y,
 const size_t incY,
 bool positive)
```

**13.229.2.12 fdot\_1\_modular\_double()**

```
double fdot_1_modular_double (
 const double p,
 const size_t n,
 const double * X,
 const size_t incX,
 const double * Y,
 const size_t incY,
 bool positive)
```

**13.229.2.13 fswap\_1\_modular\_double()**

```
void fswap_1_modular_double (
 const double p,
 const size_t n,
 double * X,
 const size_t incX,
 double * Y,
 const size_t incY,
 bool positive)
```

**13.229.2.14 fadd\_1\_modular\_double()**

```
void fadd_1_modular_double (
 const double p,
 const size_t n,
 const double * A,
 const size_t incA,
 const double * B,
 const size_t incB,
 double * C,
 const size_t incC,
 bool positive)
```

**13.229.2.15 fsub\_1\_modular\_double()**

```
void fsub_1_modular_double (
 const double p,
 const size_t n,
 const double * A,
 const size_t incA,
 const double * B,
 const size_t incB,
 double * C,
 const size_t incC,
 bool positive)
```

**13.229.2.16 faddin\_1\_modular\_double()**

```
void faddin_1_modular_double (
 const double p,
 const size_t n,
 const double * B,
 const size_t incB,
 double * C,
 const size_t incC,
 bool positive)
```

**13.229.2.17 fsubin\_1\_modular\_double()**

```
void fsubin_1_modular_double (
 const double p,
 const size_t n,
 const double * B,
 const size_t incB,
 double * C,
 const size_t incC,
 bool positive)
```

**13.230 fflas\_lvl2.C File Reference**

C functions calls for level 2 [FFLAS](#) in fflas-c.h.

```
#include "fflas-ffpack/interfaces/libs/fflas_c.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "givaro//modular-balanced.h"
#include "givaro//modular.h"
```

**Functions**

- void [fassign\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*A, const size\_t lda, double \*B, const size\_t ldb, bool positive)
- void [fzero\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, double \*A, const size\_t lda, bool positive)
- bool [fequal\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*A, const size\_t lda, const double \*B, const size\_t ldb, bool positive)
- bool [fiszero\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*A, const size\_t lda, bool positive)
- void [fidentity\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, double \*A, const size\_t lda, const double d, bool positive)
- void [freducein\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, double \*A, const size\_t lda, bool positive)

- void [freduce\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*A, const size\_t lda, double \*B, const size\_t ldb, bool positive)
- void [fnegin\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, double \*A, const size\_t lda, bool positive)
- void [fneg\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*A, const size\_t lda, double \*B, const size\_t ldb, bool positive)
- void [fscalin\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double alpha, double \*A, const size\_t lda, bool positive)
- void [fscal\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double alpha, const double \*A, const size\_t lda, double \*B, const size\_t ldb, bool positive)
- void [faxpy\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double alpha, const double \*A, const size\_t lda, double \*B, const size\_t ldb, bool positive)
- void [fmove\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, double \*A, const size\_t lda, double \*B, const size\_t ldb, bool positive)
- void [fadd\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*A, const size\_t lda, const double \*B, const size\_t ldb, double \*C, const size\_t ldc, bool positive)
- void [fsub\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*A, const size\_t lda, const double \*B, const size\_t ldb, double \*C, const size\_t ldc, bool positive)
- void [fsubin\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*B, const size\_t ldb, double \*C, const size\_t ldc, bool positive)
- void [faddin\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double \*B, const size\_t ldb, double \*C, const size\_t ldc, bool positive)
- double \* [fgemv\\_2\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_TRANSPOSE](#) TransA, const size\_t m, const size\_t n, const double alpha, const double \*A, const size\_t lda, const double \*X, const size\_t incX, const double beta, double \*Y, const size\_t incY, bool positive)
- void [fger\\_2\\_modular\\_double](#) (const double p, const size\_t m, const size\_t n, const double alpha, const double \*X, const size\_t incX, const double \*Y, const size\_t incY, double \*A, const size\_t lda, bool positive)
- void [ftrsv\\_2\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_UPLO](#) Uplo, const enum [FFLAS\\_C\\_TRANSPOSE](#) TransA, const enum [FFLAS\\_C\\_DIAG](#) Diag, const size\_t n, const double \*A, const size\_t lda, double \*X, int incX, bool positive)

### 13.230.1 Detailed Description

C functions calls for level 2 [FFLAS](#) in [fflas-c.h](#).

Author

Brice Boyer

See also

[fflas/fflas\\_level2.inl](#)

### 13.230.2 Function Documentation

#### 13.230.2.1 fassign\_2\_modular\_double()

```
void fassign_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 const double * A,
 const size_t lda,
 double * B,
 const size_t ldb,
 bool positive)
```

**13.230.2.2 fzero\_2\_modular\_double()**

```
void fzero_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 double * A,
 const size_t lda,
 bool positive)
```

**13.230.2.3 fequal\_2\_modular\_double()**

```
bool fequal_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 const double * A,
 const size_t lda,
 const double * B,
 const size_t ldb,
 bool positive)
```

**13.230.2.4 fiszero\_2\_modular\_double()**

```
bool fiszero_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 const double * A,
 const size_t lda,
 bool positive)
```

**13.230.2.5 fidentity\_2\_modular\_double()**

```
void fidentity_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 double * A,
 const size_t lda,
 const double d,
 bool positive)
```

**13.230.2.6 freducein\_2\_modular\_double()**

```
void freducein_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 double * A,
 const size_t lda,
 bool positive)
```

**13.230.2.7 freduce\_2\_modular\_double()**

```
void freduce_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
```

```
 const double * A,
 const size_t lda,
 double * B,
 const size_t ldb,
 bool positive)
```

#### 13.230.2.8 fnegin\_2\_modular\_double()

```
void fnegin_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 double * A,
 const size_t lda,
 bool positive)
```

#### 13.230.2.9 fneg\_2\_modular\_double()

```
void fneg_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 const double * A,
 const size_t lda,
 double * B,
 const size_t ldb,
 bool positive)
```

#### 13.230.2.10 fscaln\_2\_modular\_double()

```
void fscaln_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 const double alpha,
 double * A,
 const size_t lda,
 bool positive)
```

#### 13.230.2.11 fscal\_2\_modular\_double()

```
void fscal_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 const double alpha,
 const double * A,
 const size_t lda,
 double * B,
 const size_t ldb,
 bool positive)
```

#### 13.230.2.12 faxpy\_2\_modular\_double()

```
void faxpy_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 const double alpha,
```



```
 const double * A,
 const size_t lda,
 double * B,
 const size_t ldb,
 bool positive)
```

#### 13.230.2.13 fmove\_2\_modular\_double()

```
void fmove_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 double * A,
 const size_t lda,
 double * B,
 const size_t ldb,
 bool positive)
```

#### 13.230.2.14 fadd\_2\_modular\_double()

```
void fadd_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 const double * A,
 const size_t lda,
 const double * B,
 const size_t ldb,
 double * C,
 const size_t ldc,
 bool positive)
```

#### 13.230.2.15 fsub\_2\_modular\_double()

```
void fsub_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 const double * A,
 const size_t lda,
 const double * B,
 const size_t ldb,
 double * C,
 const size_t ldc,
 bool positive)
```

#### 13.230.2.16 fsubin\_2\_modular\_double()

```
void fsubin_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 const double * B,
 const size_t ldb,
 double * C,
 const size_t ldc,
 bool positive)
```

**13.230.2.17 faddin\_2\_modular\_double()**

```
void faddin_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 const double * B,
 const size_t ldb,
 double * C,
 const size_t ldc,
 bool positive)
```

**13.230.2.18 fgemv\_2\_modular\_double()**

```
double * fgemv_2_modular_double (
 const double p,
 const enum FFLAS_C_TRANSPOSE TransA,
 const size_t m,
 const size_t n,
 const double alpha,
 const double * A,
 const size_t lda,
 const double * X,
 const size_t incX,
 const double beta,
 double * Y,
 const size_t incY,
 bool positive)
```

**13.230.2.19 fger\_2\_modular\_double()**

```
void fger_2_modular_double (
 const double p,
 const size_t m,
 const size_t n,
 const double alpha,
 const double * X,
 const size_t incX,
 const double * Y,
 const size_t incY,
 double * A,
 const size_t lda,
 bool positive)
```

**13.230.2.20 ftrsv\_2\_modular\_double()**

```
void ftrsv_2_modular_double (
 const double p,
 const enum FFLAS_C_UPLO Uplo,
 const enum FFLAS_C_TRANSPOSE TransA,
 const enum FFLAS_C_DIAG Diag,
 const size_t n,
 const double * A,
 const size_t lda,
 double * X,
 int incX,
 bool positive)
```

## 13.231 fflas\_lvl3.C File Reference

C functions calls for level 3 [FFLAS](#) in fflas-c.h.

```
#include "fflas-ffpack/interfaces/libs/fflas_c.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "givaro//modular-balanced.h"
#include "givaro//modular.h"
```

### Functions

- void [ftrsm\\_3\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_SIDE](#) Side, const enum [FFLAS\\_C\\_UPLO](#) Uplo, const enum [FFLAS\\_C\\_TRANSPOSE](#) tA, const enum [FFLAS\\_C\\_DIAG](#) Diag, const size\_t m, const size\_t n, const double alpha, const double \*A, const size\_t ldA, double \*B, const size\_t ldB, bool positive)
- void [ftrmm\\_3\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_SIDE](#) Side, const enum [FFLAS\\_C\\_UPLO](#) Uplo, const enum [FFLAS\\_C\\_TRANSPOSE](#) tA, const enum [FFLAS\\_C\\_DIAG](#) Diag, const size\_t m, const size\_t n, const double alpha, double \*A, const size\_t ldA, double \*B, const size\_t ldB, bool positive)
- double \* [fgemm\\_3\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_TRANSPOSE](#) tA, const enum [FFLAS\\_C\\_TRANSPOSE](#) tB, const size\_t m, const size\_t n, const size\_t k, const double alpha, const double \*A, const size\_t ldA, const double \*B, const size\_t ldB, const double betA, double \*C, const size\_t ldC, bool positive)
- double \* [fsquare\\_3\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_TRANSPOSE](#) tA, const size\_t n, const double alpha, const double \*A, const size\_t ldA, const double betA, double \*C, const size\_t ldC, bool positive)

### 13.231.1 Detailed Description

C functions calls for level 3 [FFLAS](#) in fflas-c.h.

Author

Brice Boyer

See also

[fflas/fflas\\_level3.inl](#)

### 13.231.2 Function Documentation

#### 13.231.2.1 ftrsm\_3\_modular\_double()

```
void ftrsm_3_modular_double (
 const double p,
 const enum FFLAS_C_SIDE Side,
 const enum FFLAS_C_UPLO Uplo,
 const enum FFLAS_C_TRANSPOSE tA,
 const enum FFLAS_C_DIAG Diag,
 const size_t m,
 const size_t n,
 const double alpha,
 const double * A,
 const size_t ldA,
 double * B,
 const size_t ldB,
 bool positive)
```

### 13.231.2.2 ftrmm\_3\_modular\_double()

```
void ftrmm_3_modular_double (
 const double p,
 const enum FFLAS_C_SIDE Side,
 const enum FFLAS_C_UPLO Uplo,
 const enum FFLAS_C_TRANSPOSE tA,
 const enum FFLAS_C_DIAG Diag,
 const size_t m,
 const size_t n,
 const double alpha,
 double * A,
 const size_t ldA,
 double * B,
 const size_t ldB,
 bool positive)
```

### 13.231.2.3 fgemm\_3\_modular\_double()

```
double * fgemm_3_modular_double (
 const double p,
 const enum FFLAS_C_TRANSPOSE tA,
 const enum FFLAS_C_TRANSPOSE tB,
 const size_t m,
 const size_t n,
 const size_t k,
 const double alpha,
 const double * A,
 const size_t ldA,
 const double * B,
 const size_t ldB,
 const double betaA,
 double * C,
 const size_t ldC,
 bool positive)
```

### 13.231.2.4 fsquare\_3\_modular\_double()

```
double * fsquare_3_modular_double (
 const double p,
 const enum FFLAS_C_TRANSPOSE tA,
 const size_t n,
 const double alpha,
 const double * A,
 const size_t ldA,
 const double betaA,
 double * C,
 const size_t ldC,
 bool positive)
```

## 13.232 fflas\_sparse.C File Reference

C functions calls for level 1.5 and 2.5 [FFLAS](#) in fflas-c.h.

### 13.232.1 Detailed Description

C functions calls for level 1.5 and 2.5 [FFLAS](#) in fflas-c.h.

## Author

Brice Boyer

## See also

[fflas/fflas\\_sparse.h](#)

## 13.233 ffpack.C File Reference

C functions calls for [FFPACK](#) in `ffpack-c.h`.

```
#include "fflas-ffpack/interfaces/libs/ffpack_c.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include "givaro//modular-balanced.h"
#include "givaro//modular.h"
```

### Functions

- void [LAPACKPerm2MathPerm](#) (size\_t \*MathP, const size\_t \*LapackP, const size\_t N)
- void [MathPerm2LAPACKPerm](#) (size\_t \*LapackP, const size\_t \*MathP, const size\_t N)
- void [MatrixApplyS\\_modular\\_double](#) (const double p, double \*A, const size\_t lda, const size\_t width, const size\_t M2, const size\_t R1, const size\_t R2, const size\_t R3, const size\_t R4, bool positive)
- void [PermApplyS\\_double](#) (double \*A, const size\_t lda, const size\_t width, const size\_t M2, const size\_t R1, const size\_t R2, const size\_t R3, const size\_t R4)
- void [MatrixApplyT\\_modular\\_double](#) (const double p, double \*A, const size\_t lda, const size\_t width, const size\_t N2, const size\_t R1, const size\_t R2, const size\_t R3, const size\_t R4, bool positive)
- void [PermApplyT\\_double](#) (double \*A, const size\_t lda, const size\_t width, const size\_t N2, const size\_t R1, const size\_t R2, const size\_t R3, const size\_t R4)
- void [composePermutationsLLM](#) (size\_t \*MathP, const size\_t \*P1, const size\_t \*P2, const size\_t R, const size\_t N)
- void [composePermutationsLLL](#) (size\_t \*P1, const size\_t \*P2, const size\_t R, const size\_t N)
- void [composePermutationsMLM](#) (size\_t \*MathP1, const size\_t \*P2, const size\_t R, const size\_t N)
- void [cyclic\\_shift\\_mathPerm](#) (size\_t \*P, const size\_t s)
- void [cyclic\\_shift\\_row\\_modular\\_double](#) (const double p, double \*A, size\_t m, size\_t n, size\_t lda, bool positive)
- void [cyclic\\_shift\\_col\\_modular\\_double](#) (const double p, double \*A, size\_t m, size\_t n, size\_t lda, bool positive)
- void [applyP\\_modular\\_double](#) (const double p, const enum [FFLAS::FFLAS\\_SIDE](#) Side, const enum [FFLAS::FFLAS\\_TRANSPOSE](#) Trans, const size\_t M, const size\_t ibeg, const size\_t iend, double \*A, const size\_t lda, const size\_t \*P, bool positive)
- void [fgetrsin\\_modular\\_double](#) (const double p, const enum [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, const size\_t R, double \*A, const size\_t lda, const size\_t \*P, const size\_t \*Q, double \*B, const size\_t ldb, int \*info, bool positive)
- double \* [fgetrsv\\_modular\\_double](#) (const double p, const enum [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, const size\_t NRHS, const size\_t R, double \*A, const size\_t lda, const size\_t \*P, const size\_t \*Q, double \*X, const size\_t idx, const double \*B, const size\_t ldb, int \*info, bool positive)
- size\_t [fgesvin\\_modular\\_double](#) (const double p, const enum [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, double \*A, const size\_t lda, double \*B, const size\_t ldb, int \*info, bool positive)
- size\_t [fgesv\\_modular\\_double](#) (const double p, const enum [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, const size\_t NRHS, double \*A, const size\_t lda, double \*X, const size\_t idx, const double \*B, const size\_t ldb, int \*info, bool positive)
- void [ftrtri\\_modular\\_double](#) (const double p, const enum [FFLAS::FFLAS\\_UPLO](#) Uplo, const enum [FFLAS::FFLAS\\_DIAG](#) Diag, const size\_t N, double \*A, const size\_t lda, bool positive)
- void [trinv\\_left\\_modular\\_double](#) (const double p, const size\_t N, const double \*L, const size\_t ldl, double \*X, const size\_t idx, bool positive)
- void [ftrtrm\\_modular\\_double](#) (const double p, const [FFLAS::FFLAS\\_SIDE](#) side, const enum [FFLAS::FFLAS\\_DIAG](#) Diag, const size\_t N, double \*A, const size\_t lda, bool positive)



- `size_t pReducedColumnEchelonForm_modular_int32_t` (const int32\_t p, const size\_t M, const size\_t N, int32\_t \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)
- `size_t pReducedRowEchelonForm_modular_int32_t` (const int32\_t p, const size\_t M, const size\_t N, int32\_t \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)
- `double * Invertin_modular_double` (const double p, const size\_t M, double \*A, const size\_t lda, int \*nullity, bool positive)
- `double * Invert_modular_double` (const double p, const size\_t M, const double \*A, const size\_t lda, double \*X, const size\_t idx, int \*nullity, bool positive)
- `double * Invert2_modular_double` (const double p, const size\_t M, double \*A, const size\_t lda, double \*X, const size\_t idx, int \*nullity, bool positive)
- `size_t KrylovElim_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*P, size\_t \*Q, const size\_t deg, size\_t \*iterates, size\_t \*inviterates, const size\_t maxit, size\_t virt, bool positive)
- `size_t SpecRankProfile_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, const size\_t deg, size\_t \*rankProfile, bool positive)
- `size_t Rank_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, bool positive)
- `bool IsSingular_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, bool positive)
- `double Det_modular_double` (const double p, const size\_t N, double \*A, const size\_t lda, bool positive)
- `double * Solve_modular_double` (const double p, const size\_t M, double \*A, const size\_t lda, double \*x, const int incx, const double \*b, const int incb, bool positive)
- `void solveLB_modular_double` (const double p, const enum [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, const size\_t R, double \*L, const size\_t ldl, const size\_t \*Q, double \*B, const size\_t ldb, bool positive)
- `void solveLB2_modular_double` (const double p, const enum [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, const size\_t R, double \*L, const size\_t ldl, const size\_t \*Q, double \*B, const size\_t ldb, bool positive)
- `void RandomNullSpaceVector_modular_double` (const double p, const enum [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, double \*A, const size\_t lda, double \*X, const size\_t incX, bool positive)
- `size_t NullSpaceBasis_modular_double` (const double p, const enum [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, double \*A, const size\_t lda, double \*\*NS, size\_t \*ldn, size\_t \*NSdim, bool positive)
- `size_t RowRankProfile_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*\*rkprofile, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)
- `size_t ColumnRankProfile_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*\*rkprofile, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)
- `void RankProfileFromLU` (const size\_t \*P, const size\_t N, const size\_t R, size\_t \*rkprofile, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag)
- `size_t LeadingSubmatrixRankProfiles` (const size\_t M, const size\_t N, const size\_t R, const size\_t LSm, const size\_t LSn, const size\_t \*P, const size\_t \*Q, size\_t \*RRP, size\_t \*CRP)
- `size_t RowRankProfileSubmatrixIndices_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*\*rowindices, size\_t \*\*colindices, size\_t \*R, bool positive)
- `size_t ColRankProfileSubmatrixIndices_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*\*rowindices, size\_t \*\*colindices, size\_t \*R, bool positive)
- `size_t RowRankProfileSubmatrix_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, double \*\*X, size\_t \*R, bool positive)
- `size_t ColRankProfileSubmatrix_modular_double` (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, double \*\*X, size\_t \*R, bool positive)
- `void getTriangular_modular_double` (const double p, const enum [FFLAS::FFLAS\\_UPLO](#) Uplo, const enum [FFLAS::FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, const size\_t R, const double \*A, const size\_t lda, double \*T, const size\_t ldt, const bool OnlyNonZeroVectors, bool positive)
- `void getTriangularin_modular_double` (const double p, const enum [FFLAS::FFLAS\\_UPLO](#) Uplo, const enum [FFLAS::FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, const size\_t R, double \*A, const size\_t lda, bool positive)

- void [getEchelonForm\\_modular\\_double](#) (const double p, const enum [FFLAS::FFLAS\\_UPLO](#) Uplo, const enum [FFLAS::FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, const double \*A, const size\_t lda, double \*T, const size\_t ldt, const bool OnlyNonZeroVectors, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)
- void [getEchelonFormin\\_modular\\_double](#) (const double p, const enum [FFLAS::FFLAS\\_UPLO](#) Uplo, const enum [FFLAS::FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, double \*A, const size\_t lda, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)
- void [getEchelonTransform\\_modular\\_double](#) (const double p, const enum [FFLAS::FFLAS\\_UPLO](#) Uplo, const enum [FFLAS::FFLAS\\_DIAG](#) Diag, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, const size\_t \*Q, const double \*A, const size\_t lda, double \*T, const size\_t ldt, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)
- void [getReducedEchelonForm\\_modular\\_double](#) (const double p, const enum [FFLAS::FFLAS\\_UPLO](#) Uplo, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, const double \*A, const size\_t lda, double \*T, const size\_t ldt, const bool OnlyNonZeroVectors, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)
- void [getReducedEchelonFormin\\_modular\\_double](#) (const double p, const enum [FFLAS::FFLAS\\_UPLO](#) Uplo, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, double \*A, const size\_t lda, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)
- void [getReducedEchelonTransform\\_modular\\_double](#) (const double p, const enum [FFLAS::FFLAS\\_UPLO](#) Uplo, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, const size\_t \*Q, const double \*A, const size\_t lda, double \*T, const size\_t ldt, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)
- void [PLUQtoEchelonPermutation](#) (const size\_t N, const size\_t R, const size\_t \*P, size\_t \*outPerm)

### 13.233.1 Detailed Description

C functions calls for [FFPACK](#) in `ffpack-c.h`.

Author

Brice Boyer

See also

[ffpack/ffpack.h](#)

### 13.233.2 Function Documentation

#### 13.233.2.1 LAPACKPerm2MathPerm()

```
void LAPACKPerm2MathPerm (
 size_t * MathP,
 const size_t * LapackP,
 const size_t N)
```

#### 13.233.2.2 MathPerm2LAPACKPerm()

```
void MathPerm2LAPACKPerm (
 size_t * LapackP,
 const size_t * MathP,
 const size_t N)
```

#### 13.233.2.3 MatrixApplyS\_modular\_double()

```
void MatrixApplyS_modular_double (
 const double p,
 double * A,
 const size_t lda,
 const size_t width,
 const size_t M2,
 const size_t R1,
 const size_t R2,
```



```
 const size_t R3,
 const size_t R4,
 bool positive)
```

#### 13.233.2.4 PermApplyS\_double()

```
void PermApplyS_double (
 double * A,
 const size_t lda,
 const size_t width,
 const size_t M2,
 const size_t R1,
 const size_t R2,
 const size_t R3,
 const size_t R4)
```

#### 13.233.2.5 MatrixApplyT\_modular\_double()

```
void MatrixApplyT_modular_double (
 const double p,
 double * A,
 const size_t lda,
 const size_t width,
 const size_t N2,
 const size_t R1,
 const size_t R2,
 const size_t R3,
 const size_t R4,
 bool positive)
```

#### 13.233.2.6 PermApplyT\_double()

```
void PermApplyT_double (
 double * A,
 const size_t lda,
 const size_t width,
 const size_t N2,
 const size_t R1,
 const size_t R2,
 const size_t R3,
 const size_t R4)
```

#### 13.233.2.7 composePermutationsLLM()

```
void composePermutationsLLM (
 size_t * MathP,
 const size_t * P1,
 const size_t * P2,
 const size_t R,
 const size_t N)
```

#### 13.233.2.8 composePermutationsLLL()

```
void composePermutationsLLL (
 size_t * P1,
 const size_t * P2,
 const size_t R,
 const size_t N)
```

**13.233.2.9 composePermutationsMLM()**

```
void composePermutationsMLM (
 size_t * MathP1,
 const size_t * P2,
 const size_t R,
 const size_t N)
```

**13.233.2.10 cyclic\_shift\_mathPerm()**

```
void cyclic_shift_mathPerm (
 size_t * P,
 const size_t s)
```

**13.233.2.11 cyclic\_shift\_row\_modular\_double()**

```
void cyclic_shift_row_modular_double (
 const double p,
 double * A,
 size_t m,
 size_t n,
 size_t lda,
 bool positive)
```

**13.233.2.12 cyclic\_shift\_col\_modular\_double()**

```
void cyclic_shift_col_modular_double (
 const double p,
 double * A,
 size_t m,
 size_t n,
 size_t lda,
 bool positive)
```

**13.233.2.13 applyP\_modular\_double()**

```
void applyP_modular_double (
 const double p,
 const enum FFLAS::FFLAS_SIDE Side,
 const enum FFLAS::FFLAS_TRANSPOSE Trans,
 const size_t M,
 const size_t ibeg,
 const size_t iend,
 double * A,
 const size_t lda,
 const size_t * P,
 bool positive)
```

**13.233.2.14 fgetrsin\_modular\_double()**

```
void fgetrsin_modular_double (
 const double p,
 const enum FFLAS::FFLAS_SIDE Side,
 const size_t M,
 const size_t N,
 const size_t R,
 double * A,
 const size_t lda,
 const size_t * P,
```

```
 const size_t * Q,
 double * B,
 const size_t ldb,
 int * info,
 bool positive)
```

#### 13.233.2.15 fgetrsv\_modular\_double()

```
double * fgetrsv_modular_double (
 const double p,
 const enum FFLAS::FFLAS_SIDE Side,
 const size_t M,
 const size_t N,
 const size_t NRHS,
 const size_t R,
 double * A,
 const size_t lda,
 const size_t * P,
 const size_t * Q,
 double * X,
 const size_t ldx,
 const double * B,
 const size_t ldb,
 int * info,
 bool positive)
```

#### 13.233.2.16 fgesvin\_modular\_double()

```
size_t fgesvin_modular_double (
 const double p,
 const enum FFLAS::FFLAS_SIDE Side,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 double * B,
 const size_t ldb,
 int * info,
 bool positive)
```

#### 13.233.2.17 fgesv\_modular\_double()

```
size_t fgesv_modular_double (
 const double p,
 const enum FFLAS::FFLAS_SIDE Side,
 const size_t M,
 const size_t N,
 const size_t NRHS,
 double * A,
 const size_t lda,
 double * X,
 const size_t ldx,
 const double * B,
 const size_t ldb,
 int * info,
 bool positive)
```

**13.233.2.18 ftrtri\_modular\_double()**

```
void ftrtri_modular_double (
 const double p,
 const enum FFLAS::FFLAS_UPLO Uplo,
 const enum FFLAS::FFLAS_DIAG Diag,
 const size_t N,
 double * A,
 const size_t lda,
 bool positive)
```

**13.233.2.19 trinv\_left\_modular\_double()**

```
void trinv_left_modular_double (
 const double p,
 const size_t N,
 const double * L,
 const size_t ldl,
 double * X,
 const size_t ldx,
 bool positive)
```

**13.233.2.20 ftrtrm\_modular\_double()**

```
void ftrtrm_modular_double (
 const double p,
 const FFLAS::FFLAS_SIDE side,
 const enum FFLAS::FFLAS_DIAG Diag,
 const size_t N,
 double * A,
 const size_t lda,
 bool positive)
```

**13.233.2.21 PLUQ\_modular\_double()**

```
size_t PLUQ_modular_double (
 const double p,
 const enum FFLAS::FFLAS_DIAG Diag,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 size_t * P,
 size_t * Q,
 bool positive)
```

**13.233.2.22 LUdivine\_modular\_double()**

```
size_t LUdivine_modular_double (
 const double p,
 const enum FFLAS::FFLAS_DIAG Diag,
 const enum FFLAS::FFLAS_TRANSPOSE Trans,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const enum FFPACK_C_LU_TAG LuTag,
```

```

 const size_t cutoff,
 bool positive)

```

#### 13.233.2.23 ColumnEchelonForm\_modular\_double()

```

size_t ColumnEchelonForm_modular_double (
 const double p,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)

```

#### 13.233.2.24 RowEchelonForm\_modular\_double()

```

size_t RowEchelonForm_modular_double (
 const double p,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)

```

#### 13.233.2.25 ReducedColumnEchelonForm\_modular\_double()

```

size_t ReducedColumnEchelonForm_modular_double (
 const double p,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)

```

#### 13.233.2.26 ReducedRowEchelonForm\_modular\_double()

```

size_t ReducedRowEchelonForm_modular_double (
 const double p,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)

```

**13.233.2.27 ColumnEchelonForm\_modular\_float()**

```
size_t ColumnEchelonForm_modular_float (
 const float p,
 const size_t M,
 const size_t N,
 float * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)
```

**13.233.2.28 RowEchelonForm\_modular\_float()**

```
size_t RowEchelonForm_modular_float (
 const float p,
 const size_t M,
 const size_t N,
 float * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)
```

**13.233.2.29 ReducedColumnEchelonForm\_modular\_float()**

```
size_t ReducedColumnEchelonForm_modular_float (
 const float p,
 const size_t M,
 const size_t N,
 float * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)
```

**13.233.2.30 ReducedRowEchelonForm\_modular\_float()**

```
size_t ReducedRowEchelonForm_modular_float (
 const float p,
 const size_t M,
 const size_t N,
 float * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)
```

**13.233.2.31 ColumnEchelonForm\_modular\_int32\_t()**

```
size_t ColumnEchelonForm_modular_int32_t (
 const int32_t p,
```

```

 const size_t M,
 const size_t N,
 int32_t * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)

```

#### 13.233.2.32 RowEchelonForm\_modular\_int32\_t()

```

size_t RowEchelonForm_modular_int32_t (
 const int32_t p,
 const size_t M,
 const size_t N,
 int32_t * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)

```

#### 13.233.2.33 ReducedColumnEchelonForm\_modular\_int32\_t()

```

size_t ReducedColumnEchelonForm_modular_int32_t (
 const int32_t p,
 const size_t M,
 const size_t N,
 int32_t * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)

```

#### 13.233.2.34 ReducedRowEchelonForm\_modular\_int32\_t()

```

size_t ReducedRowEchelonForm_modular_int32_t (
 const int32_t p,
 const size_t M,
 const size_t N,
 int32_t * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)

```

#### 13.233.2.35 pColumnEchelonForm\_modular\_double()

```

size_t pColumnEchelonForm_modular_double (
 const double p,
 const size_t M,
 const size_t N,
 double * A,

```

```

 const size_t lda,
 size_t * P,
 size_t * Qt,
 bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)

```

### 13.233.2.36 pRowEchelonForm\_modular\_double()

```

size_t pRowEchelonForm_modular_double (
 const double p,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)

```

### 13.233.2.37 pReducedColumnEchelonForm\_modular\_double()

```

size_t pReducedColumnEchelonForm_modular_double (
 const double p,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)

```

### 13.233.2.38 pReducedRowEchelonForm\_modular\_double()

```

size_t pReducedRowEchelonForm_modular_double (
 const double p,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)

```

### 13.233.2.39 pColumnEchelonForm\_modular\_float()

```

size_t pColumnEchelonForm_modular_float (
 const float p,
 const size_t M,
 const size_t N,
 float * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,

```



```

 bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)

```

#### 13.233.2.40 pRowEchelonForm\_modular\_float()

```

size_t pRowEchelonForm_modular_float (
 const float p,
 const size_t M,
 const size_t N,
 float * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)

```

#### 13.233.2.41 pReducedColumnEchelonForm\_modular\_float()

```

size_t pReducedColumnEchelonForm_modular_float (
 const float p,
 const size_t M,
 const size_t N,
 float * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)

```

#### 13.233.2.42 pReducedRowEchelonForm\_modular\_float()

```

size_t pReducedRowEchelonForm_modular_float (
 const float p,
 const size_t M,
 const size_t N,
 float * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)

```

#### 13.233.2.43 pColumnEchelonForm\_modular\_int32\_t()

```

size_t pColumnEchelonForm_modular_int32_t (
 const int32_t p,
 const size_t M,
 const size_t N,
 int32_t * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)

```

**13.233.2.44 pRowEchelonForm\_modular\_int32\_t()**

```
size_t pRowEchelonForm_modular_int32_t (
 const int32_t p,
 const size_t M,
 const size_t N,
 int32_t * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)
```

**13.233.2.45 pReducedColumnEchelonForm\_modular\_int32\_t()**

```
size_t pReducedColumnEchelonForm_modular_int32_t (
 const int32_t p,
 const size_t M,
 const size_t N,
 int32_t * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)
```

**13.233.2.46 pReducedRowEchelonForm\_modular\_int32\_t()**

```
size_t pReducedRowEchelonForm_modular_int32_t (
 const int32_t p,
 const size_t M,
 const size_t N,
 int32_t * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)
```

**13.233.2.47 Invertin\_modular\_double()**

```
double * Invertin_modular_double (
 const double p,
 const size_t M,
 double * A,
 const size_t lda,
 int * nullity,
 bool positive)
```

**13.233.2.48 Invert\_modular\_double()**

```
double * Invert_modular_double (
 const double p,
 const size_t M,
 const double * A,
 const size_t lda,
```

```
double * X,
const size_t ldx,
int * nullity,
bool positive)
```

#### 13.233.2.49 Invert2\_modular\_double()

```
double * Invert2_modular_double (
 const double p,
 const size_t M,
 double * A,
 const size_t lda,
 double * X,
 const size_t ldx,
 int * nullity,
 bool positive)
```

#### 13.233.2.50 KrylovElim\_modular\_double()

```
size_t KrylovElim_modular_double (
 const double p,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 size_t * P,
 size_t * Q,
 const size_t deg,
 size_t * iterates,
 size_t * inviterates,
 const size_t maxit,
 size_t virt,
 bool positive)
```

#### 13.233.2.51 SpecRankProfile\_modular\_double()

```
size_t SpecRankProfile_modular_double (
 const double p,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 const size_t deg,
 size_t * rankProfile,
 bool positive)
```

#### 13.233.2.52 Rank\_modular\_double()

```
size_t Rank_modular_double (
 const double p,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 bool positive)
```

#### 13.233.2.53 IsSingular\_modular\_double()

```
bool IsSingular_modular_double (

```

```

 const double p,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 bool positive)

```

#### 13.233.2.54 Det\_modular\_double()

```

double Det_modular_double (
 const double p,
 const size_t N,
 double * A,
 const size_t lda,
 bool positive)

```

#### 13.233.2.55 Solve\_modular\_double()

```

double * Solve_modular_double (
 const double p,
 const size_t M,
 double * A,
 const size_t lda,
 double * x,
 const int incx,
 const double * b,
 const int incb,
 bool positive)

```

#### 13.233.2.56 solveLB\_modular\_double()

```

void solveLB_modular_double (
 const double p,
 const enum FFLAS::FFLAS_SIDE Side,
 const size_t M,
 const size_t N,
 const size_t R,
 double * L,
 const size_t ldl,
 const size_t * Q,
 double * B,
 const size_t ldb,
 bool positive)

```

#### 13.233.2.57 solveLB2\_modular\_double()

```

void solveLB2_modular_double (
 const double p,
 const enum FFLAS::FFLAS_SIDE Side,
 const size_t M,
 const size_t N,
 const size_t R,
 double * L,
 const size_t ldl,
 const size_t * Q,
 double * B,
 const size_t ldb,
 bool positive)

```

**13.233.2.58 RandomNullSpaceVector\_modular\_double()**

```
void RandomNullSpaceVector_modular_double (
 const double p,
 const enum FFLAS::FFLAS_SIDE Side,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 double * X,
 const size_t incX,
 bool positive)
```

**13.233.2.59 NullSpaceBasis\_modular\_double()**

```
size_t NullSpaceBasis_modular_double (
 const double p,
 const enum FFLAS::FFLAS_SIDE Side,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 double ** NS,
 size_t * ldn,
 size_t * NSdim,
 bool positive)
```

**13.233.2.60 RowRankProfile\_modular\_double()**

```
size_t RowRankProfile_modular_double (
 const double p,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 size_t ** rkprofile,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)
```

**13.233.2.61 ColumnRankProfile\_modular\_double()**

```
size_t ColumnRankProfile_modular_double (
 const double p,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 size_t ** rkprofile,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)
```

**13.233.2.62 RankProfileFromLU()**

```
void RankProfileFromLU (
 const size_t * P,
 const size_t N,
 const size_t R,
 size_t * rkprofile,
 const enum FFPACK_C_LU_TAG LuTag)
```

**13.233.2.63 LeadingSubmatrixRankProfiles()**

```
size_t LeadingSubmatrixRankProfiles (
 const size_t M,
 const size_t N,
 const size_t R,
 const size_t LSm,
 const size_t LSn,
 const size_t * P,
 const size_t * Q,
 size_t * RRP,
 size_t * CRP)
```

**13.233.2.64 RowRankProfileSubmatrixIndices\_modular\_double()**

```
size_t RowRankProfileSubmatrixIndices_modular_double (
 const double p,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 size_t ** rowindices,
 size_t ** colindices,
 size_t * R,
 bool positive)
```

**13.233.2.65 ColRankProfileSubmatrixIndices\_modular\_double()**

```
size_t ColRankProfileSubmatrixIndices_modular_double (
 const double p,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 size_t ** rowindices,
 size_t ** colindices,
 size_t * R,
 bool positive)
```

**13.233.2.66 RowRankProfileSubmatrix\_modular\_double()**

```
size_t RowRankProfileSubmatrix_modular_double (
 const double p,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 double ** X,
 size_t * R,
 bool positive)
```

**13.233.2.67 ColRankProfileSubmatrix\_modular\_double()**

```
size_t ColRankProfileSubmatrix_modular_double (
 const double p,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
```

```
double ** X,
size_t * R,
bool positive)
```

### 13.233.2.68 getTriangular\_modular\_double()

```
void getTriangular_modular_double (
 const double p,
 const enum FFLAS::FFLAS_UPLO Uplo,
 const enum FFLAS::FFLAS_DIAG Diag,
 const size_t M,
 const size_t N,
 const size_t R,
 const double * A,
 const size_t lda,
 double * T,
 const size_t ldt,
 const bool OnlyNonZeroVectors,
 bool positive)
```

### 13.233.2.69 getTriangularin\_modular\_double()

```
void getTriangularin_modular_double (
 const double p,
 const enum FFLAS::FFLAS_UPLO Uplo,
 const enum FFLAS::FFLAS_DIAG Diag,
 const size_t M,
 const size_t N,
 const size_t R,
 double * A,
 const size_t lda,
 bool positive)
```

### 13.233.2.70 getEchelonForm\_modular\_double()

```
void getEchelonForm_modular_double (
 const double p,
 const enum FFLAS::FFLAS_UPLO Uplo,
 const enum FFLAS::FFLAS_DIAG Diag,
 const size_t M,
 const size_t N,
 const size_t R,
 const size_t * P,
 const double * A,
 const size_t lda,
 double * T,
 const size_t ldt,
 const bool OnlyNonZeroVectors,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)
```

### 13.233.2.71 getEchelonFormin\_modular\_double()

```
void getEchelonFormin_modular_double (
 const double p,
 const enum FFLAS::FFLAS_UPLO Uplo,
 const enum FFLAS::FFLAS_DIAG Diag,
 const size_t M,
 const size_t N,
```

```

 const size_t R,
 const size_t * P,
 double * A,
 const size_t lda,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)

```

### 13.233.2.72 getEchelonTransform\_modular\_double()

```

void getEchelonTransform_modular_double (
 const double p,
 const enum FFLAS::FFLAS_UPLO Uplo,
 const enum FFLAS::FFLAS_DIAG Diag,
 const size_t M,
 const size_t N,
 const size_t R,
 const size_t * P,
 const size_t * Q,
 const double * A,
 const size_t lda,
 double * T,
 const size_t ldt,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)

```

### 13.233.2.73 getReducedEchelonForm\_modular\_double()

```

void getReducedEchelonForm_modular_double (
 const double p,
 const enum FFLAS::FFLAS_UPLO Uplo,
 const size_t M,
 const size_t N,
 const size_t R,
 const size_t * P,
 const double * A,
 const size_t lda,
 double * T,
 const size_t ldt,
 const bool OnlyNonZeroVectors,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)

```

### 13.233.2.74 getReducedEchelonFormin\_modular\_double()

```

void getReducedEchelonFormin_modular_double (
 const double p,
 const enum FFLAS::FFLAS_UPLO Uplo,
 const size_t M,
 const size_t N,
 const size_t R,
 const size_t * P,
 double * A,
 const size_t lda,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)

```

### 13.233.2.75 getReducedEchelonTransform\_modular\_double()

```

void getReducedEchelonTransform_modular_double (

```



```

const double p,
const enum FFLAS::FFLAS_UPLO Uplo,
const size_t M,
const size_t N,
const size_t R,
const size_t * P,
const size_t * Q,
const double * A,
const size_t lda,
double * T,
const size_t ldt,
const enum FFPACK_C_LU_TAG LuTag,
bool positive)

```

### 13.233.2.76 PLUQtoEchelonPermutation()

```

void PLUQtoEchelonPermutation (
 const size_t N,
 const size_t R,
 const size_t * P,
 size_t * outPerm)

```

## 13.234 ffpack\_c.h File Reference

```

#include <stdbool.h>
#include <stdlib.h>
#include <inttypes.h>

```

### Macros

- #define [FFPACK\\_COMPILED](#)

### Enumerations

- enum [FFLAS\\_C\\_ORDER](#) { [FflasRowMajor](#) =101 , [FflasColMajor](#) =102 }
- enum [FFLAS\\_C\\_TRANSPOSE](#) { [FflasNoTrans](#) = 111 , [FflasTrans](#) = 112 }
- enum [FFLAS\\_C\\_UPLO](#) { [FflasUpper](#) = 121 , [FflasLower](#) = 122 }
- enum [FFLAS\\_C\\_DIAG](#) { [FflasNonUnit](#) = 131 , [FflasUnit](#) = 132 }
- enum [FFLAS\\_C\\_SIDE](#) { [FflasLeft](#) = 141 , [FflasRight](#) = 142 }
- enum [FFPACK\\_C\\_LU\\_TAG](#) { [FfpackSlabRecursive](#) = 1 , [FfpackTileRecursive](#) = 2 , [FfpackSingular](#) = 3 }
- enum [FFPACK\\_C\\_CHARPOLY\\_TAG](#) {  
[FfpackLUK](#) =1 , [FfpackKG](#) =2 , [FfpackHybrid](#) =3 , [FfpackKGFast](#) =4 ,  
[FfpackDanilevski](#) =5 , [FfpackArithProg](#) =6 , [FfpackKGFastG](#) =7 }
- enum [FFPACK\\_C\\_MINPOLY\\_TAG](#) { [FfpackDense](#) =1 , [FfpackKGF](#) =2 }

### Functions

- void [LAPACKPerm2MathPerm](#) (size\_t \*MathP, const size\_t \*LapackP, const size\_t N)
- void [MathPerm2LAPACKPerm](#) (size\_t \*LapackP, const size\_t \*MathP, const size\_t N)
- void [MatrixApplyS\\_modular\\_double](#) (const double p, double \*A, const size\_t lda, const size\_t width, const size\_t M2, const size\_t R1, const size\_t R2, const size\_t R3, const size\_t R4, bool positive)
- void [PermApplyS\\_double](#) (double \*A, const size\_t lda, const size\_t width, const size\_t M2, const size\_t R1, const size\_t R2, const size\_t R3, const size\_t R4)
- void [MatrixApplyT\\_modular\\_double](#) (const double p, double \*A, const size\_t lda, const size\_t width, const size\_t N2, const size\_t R1, const size\_t R2, const size\_t R3, const size\_t R4, bool positive)
- void [PermApplyT\\_double](#) (double \*A, const size\_t lda, const size\_t width, const size\_t N2, const size\_t R1, const size\_t R2, const size\_t R3, const size\_t R4)

- void [composePermutationsLLM](#) (size\_t \*MathP, const size\_t \*P1, const size\_t \*P2, const size\_t R, const size\_t N)
- void [composePermutationsLLL](#) (size\_t \*P1, const size\_t \*P2, const size\_t R, const size\_t N)
- void [composePermutationsMLM](#) (size\_t \*MathP1, const size\_t \*P2, const size\_t R, const size\_t N)
- void [cyclic\\_shift\\_mathPerm](#) (size\_t \*P, const size\_t s)
- void [cyclic\\_shift\\_row\\_modular\\_double](#) (const double p, double \*A, size\_t m, size\_t n, size\_t lda, bool positive)
- void [cyclic\\_shift\\_col\\_modular\\_double](#) (const double p, double \*A, size\_t m, size\_t n, size\_t lda, bool positive)
- void [applyP\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_SIDE](#) Side, const enum [FFLAS\\_C\\_TRANSPOSE](#) Trans, const size\_t M, const size\_t ibeg, const size\_t iend, double \*A, const size\_t lda, const size\_t \*P, bool positive)
- void [fgetrsin\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_SIDE](#) Side, const size\_t M, const size\_t N, const size\_t R, double \*A, const size\_t lda, const size\_t \*P, const size\_t \*Q, double \*B, const size\_t ldb, int \*info, bool positive)
- double \* [fgetrs\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_SIDE](#) Side, const size\_t M, const size\_t N, const size\_t NRHS, const size\_t R, double \*A, const size\_t lda, const size\_t \*P, const size\_t \*Q, double \*X, const size\_t ldx, const double \*B, const size\_t ldb, int \*info, bool positive)
- size\_t [fgesvin\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_SIDE](#) Side, const size\_t M, const size\_t N, double \*A, const size\_t lda, double \*B, const size\_t ldb, int \*info, bool positive)
- size\_t [fgesv\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_SIDE](#) Side, const size\_t M, const size\_t N, const size\_t NRHS, double \*A, const size\_t lda, double \*X, const size\_t ldx, const double \*B, const size\_t ldb, int \*info)
- void [ftrtri\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_UPLO](#) Uplo, const enum [FFLAS\\_C\\_DIAG](#) Diag, const size\_t N, double \*A, const size\_t lda, bool positive)
- void [trinv\\_left\\_modular\\_double](#) (const double p, const size\_t N, const double \*L, const size\_t ldl, double \*X, const size\_t ldx, bool positive)
- void [ftrtrm\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_DIAG](#) diag, const size\_t N, double \*A, const size\_t lda, bool positive)
- size\_t [PLUQ\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_DIAG](#) Diag, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*P, size\_t \*Q, bool positive)
- size\_t [LUdivine\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_DIAG](#) Diag, const enum [FFLAS\\_C\\_TRANSPOSE](#) trans, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, const size\_t cutoff, bool positive)
- size\_t [LUdivine\\_small\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_DIAG](#) Diag, const enum [FFLAS\\_C\\_TRANSPOSE](#) trans, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*P, size\_t \*Q, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)
- size\_t [LUdivine\\_gauss\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_DIAG](#) Diag, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*P, size\_t \*Q, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)
- size\_t [ColumnEchelonForm\\_modular\\_double](#) (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, bool transform, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)
- size\_t [RowEchelonForm\\_modular\\_double](#) (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)
- size\_t [ColumnEchelonForm\\_modular\\_float](#) (const float p, const size\_t M, const size\_t N, float \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, bool transform, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)
- size\_t [RowEchelonForm\\_modular\\_float](#) (const float p, const size\_t M, const size\_t N, float \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)
- size\_t [ColumnEchelonForm\\_modular\\_int32\\_t](#) (const int32\_t p, const size\_t M, const size\_t N, int32\_t \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, bool transform, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)
- size\_t [RowEchelonForm\\_modular\\_int32\\_t](#) (const int32\_t p, const size\_t M, const size\_t N, int32\_t \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)
- size\_t [ReducedColumnEchelonForm\\_modular\\_double](#) (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)
- size\_t [ReducedRowEchelonForm\\_modular\\_double](#) (const double p, const size\_t M, const size\_t N, double \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)

- `size_t ReducedColumnEchelonForm_modular_float` (const float p, const `size_t` M, const `size_t` N, float \*A, const `size_t` lda, `size_t` \*P, `size_t` \*Qt, const bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t ReducedRowEchelonForm_modular_float` (const float p, const `size_t` M, const `size_t` N, float \*A, const `size_t` lda, `size_t` \*P, `size_t` \*Qt, const bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t ReducedColumnEchelonForm_modular_int32_t` (const `int32_t` p, const `size_t` M, const `size_t` N, `int32_t` \*A, const `size_t` lda, `size_t` \*P, `size_t` \*Qt, const bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t ReducedRowEchelonForm_modular_int32_t` (const `int32_t` p, const `size_t` M, const `size_t` N, `int32_t` \*A, const `size_t` lda, `size_t` \*P, `size_t` \*Qt, const bool transform, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t ReducedRowEchelonForm2_modular_double` (const double p, const `size_t` M, const `size_t` N, double \*A, const `size_t` lda, `size_t` \*P, `size_t` \*Qt, const bool transform, bool positive)
- `size_t REF_modular_double` (const double p, const `size_t` M, const `size_t` N, double \*A, const `size_t` lda, const `size_t` colbeg, const `size_t` rowbeg, const `size_t` colsize, `size_t` \*Qt, `size_t` \*P, bool positive)
- `double * Invertin_modular_double` (const double p, const `size_t` M, double \*A, const `size_t` lda, int \*nullity, bool positive)
- `double * Invert_modular_double` (const double p, const `size_t` M, const double \*A, const `size_t` lda, double \*X, const `size_t` ldx, int \*nullity, bool positive)
- `double * Invert2_modular_double` (const double p, const `size_t` M, double \*A, const `size_t` lda, double \*X, const `size_t` ldx, int \*nullity, bool positive)
- `size_t KrylovElim_modular_double` (const double p, const `size_t` M, const `size_t` N, double \*A, const `size_t` lda, `size_t` \*P, `size_t` \*Q, const `size_t` deg, `size_t` \*iterates, `size_t` \*inviterates, const `size_t` maxit, `size_t` virt, bool positive)
- `size_t SpecRankProfile_modular_double` (const double p, const `size_t` M, const `size_t` N, double \*A, const `size_t` lda, const `size_t` deg, `size_t` \*rankProfile, bool positive)
- `size_t Rank_modular_double` (const double p, const `size_t` M, const `size_t` N, double \*A, const `size_t` lda, bool positive)
- `bool IsSingular_modular_double` (const double p, const `size_t` M, const `size_t` N, double \*A, const `size_t` lda, bool positive)
- `double Det_modular_double` (const double p, const `size_t` N, double \*A, const `size_t` lda, bool positive)
- `double * Solve_modular_double` (const double p, const `size_t` M, double \*A, const `size_t` lda, double \*x, const int incx, const double \*b, const int incb, bool positive)
- `void solveLB_modular_double` (const double p, const enum `FFLAS_C_SIDE` Side, const `size_t` M, const `size_t` N, const `size_t` R, double \*L, const `size_t` ldl, const `size_t` \*Q, double \*B, const `size_t` ldb)
- `void solveLB2_modular_double` (const double p, const enum `FFLAS_C_SIDE` Side, const `size_t` M, const `size_t` N, const `size_t` R, double \*L, const `size_t` ldl, const `size_t` \*Q, double \*B, const `size_t` ldb, bool positive)
- `void RandomNullSpaceVector_modular_double` (const double p, const enum `FFLAS_C_SIDE` Side, const `size_t` M, const `size_t` N, double \*A, const `size_t` lda, double \*X, const `size_t` incX, bool positive)
- `size_t NullSpaceBasis_modular_double` (const double p, const enum `FFLAS_C_SIDE` Side, const `size_t` M, const `size_t` N, double \*A, const `size_t` lda, double \*\*NS, `size_t` \*ldn, `size_t` \*NSdim, bool positive)
- `size_t RowRankProfile_modular_double` (const double p, const `size_t` M, const `size_t` N, double \*A, const `size_t` lda, `size_t` \*\*rkprofile, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `size_t ColumnRankProfile_modular_double` (const double p, const `size_t` M, const `size_t` N, double \*A, const `size_t` lda, `size_t` \*\*rkprofile, const enum `FFPACK_C_LU_TAG` LuTag, bool positive)
- `void RankProfileFromLU` (const `size_t` \*P, const `size_t` N, const `size_t` R, `size_t` \*rkprofile, const enum `FFPACK_C_LU_TAG` LuTag)
- `size_t LeadingSubmatrixRankProfiles` (const `size_t` M, const `size_t` N, const `size_t` R, const `size_t` LSm, const `size_t` LSn, const `size_t` \*P, const `size_t` \*Q, `size_t` \*RRP, `size_t` \*CRP)
- `size_t RowRankProfileSubmatrixIndices_modular_double` (const double p, const `size_t` M, const `size_t` N, double \*A, const `size_t` lda, `size_t` \*\*rowindices, `size_t` \*\*colindices, `size_t` \*R, bool positive)
- `size_t ColRankProfileSubmatrixIndices_modular_double` (const double p, const `size_t` M, const `size_t` N, double \*A, const `size_t` lda, `size_t` \*\*rowindices, `size_t` \*\*colindices, `size_t` \*R, bool positive)
- `size_t RowRankProfileSubmatrix_modular_double` (const double p, const `size_t` M, const `size_t` N, double \*A, const `size_t` lda, double \*\*X, `size_t` \*R, bool positive)
- `size_t ColRankProfileSubmatrix_modular_double` (const double p, const `size_t` M, const `size_t` N, double \*A, const `size_t` lda, double \*\*X, `size_t` \*R, bool positive)

- void [getTriangular\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_UPLO](#) Uplo, const enum [FFLAS\\_C\\_DIAG](#) diag, const size\_t M, const size\_t N, const size\_t R, const double \*A, const size\_t lda, double \*T, const size\_t ldt, const bool OnlyNonZeroVectors, bool positive)
- void [getTriangularin\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_UPLO](#) Uplo, const enum [FFLAS\\_C\\_DIAG](#) diag, const size\_t M, const size\_t N, const size\_t R, double \*A, const size\_t lda, bool positive)
- void [getEchelonForm\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_UPLO](#) Uplo, const enum [FFLAS\\_C\\_DIAG](#) diag, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, const double \*A, const size\_t lda, double \*T, const size\_t ldt, const bool OnlyNonZeroVectors, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)
- void [getEchelonFormin\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_UPLO](#) Uplo, const enum [FFLAS\\_C\\_DIAG](#) diag, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, double \*A, const size\_t lda, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)
- void [getEchelonTransform\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_UPLO](#) Uplo, const enum [FFLAS\\_C\\_DIAG](#) diag, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, const size\_t \*Q, const double \*A, const size\_t lda, double \*T, const size\_t ldt, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)
- void [getReducedEchelonForm\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_UPLO](#) Uplo, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, const double \*A, const size\_t lda, double \*T, const size\_t ldt, const bool OnlyNonZeroVectors, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)
- void [getReducedEchelonFormin\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_UPLO](#) Uplo, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, double \*A, const size\_t lda, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)
- void [getReducedEchelonTransform\\_modular\\_double](#) (const double p, const enum [FFLAS\\_C\\_UPLO](#) Uplo, const size\_t M, const size\_t N, const size\_t R, const size\_t \*P, const size\_t \*Q, const double \*A, const size\_t lda, double \*T, const size\_t ldt, const enum [FFPACK\\_C\\_LU\\_TAG](#) LuTag, bool positive)
- void [PLUQtoEchelonPermutation](#) (const size\_t N, const size\_t R, const size\_t \*P, size\_t \*outPerm)

### 13.234.1 Macro Definition Documentation

#### 13.234.1.1 FFPACK\_COMPILED

```
#define FFPACK_COMPILED
```

### 13.234.2 Enumeration Type Documentation

#### 13.234.2.1 FFLAS\_C\_ORDER

```
enum FFLAS_C_ORDER
```

Enumerator

|               |  |
|---------------|--|
| FflasRowMajor |  |
| FflasColMajor |  |

#### 13.234.2.2 FFLAS\_C\_TRANSPOSE

```
enum FFLAS_C_TRANSPOSE
```

Enumerator

|              |  |
|--------------|--|
| FflasNoTrans |  |
| FflasTrans   |  |

#### 13.234.2.3 FFLAS\_C\_UPLO

```
enum FFLAS_C_UPLO
```

## Enumerator

|            |  |
|------------|--|
| FflasUpper |  |
| FflasLower |  |

**13.234.2.4 FFLAS\_C\_DIAG**enum [FFLAS\\_C\\_DIAG](#)

## Enumerator

|              |  |
|--------------|--|
| FflasNonUnit |  |
| FflasUnit    |  |

**13.234.2.5 FFLAS\_C\_SIDE**enum [FFLAS\\_C\\_SIDE](#)

## Enumerator

|            |  |
|------------|--|
| FflasLeft  |  |
| FflasRight |  |

**13.234.2.6 FFPACK\_C\_LU\_TAG**enum [FFPACK\\_C\\_LU\\_TAG](#)

## Enumerator

|                     |  |
|---------------------|--|
| FfpackSlabRecursive |  |
| FfpackTileRecursive |  |
| FfpackSingular      |  |

**13.234.2.7 FFPACK\_C\_CHARPOLY\_TAG**enum [FFPACK\\_C\\_CHARPOLY\\_TAG](#)

## Enumerator

|                  |  |
|------------------|--|
| FfpackLUK        |  |
| FfpackKG         |  |
| FfpackHybrid     |  |
| FfpackKGFast     |  |
| FfpackDanilevski |  |
| FfpackArithProg  |  |
| FfpackKGFastG    |  |

**13.234.2.8 FFPACK\_C\_MINPOLY\_TAG**enum [FFPACK\\_C\\_MINPOLY\\_TAG](#)

## Enumerator

|             |  |
|-------------|--|
| FfpackDense |  |
| FfpackKGF   |  |

**13.234.3 Function Documentation****13.234.3.1 LAPACKPerm2MathPerm()**

```
void LAPACKPerm2MathPerm (
 size_t * MathP,
 const size_t * LapackP,
 const size_t N)
```

**13.234.3.2 MathPerm2LAPACKPerm()**

```
void MathPerm2LAPACKPerm (
 size_t * LapackP,
 const size_t * MathP,
 const size_t N)
```

**13.234.3.3 MatrixApplyS\_modular\_double()**

```
void MatrixApplyS_modular_double (
 const double p,
 double * A,
 const size_t lda,
 const size_t width,
 const size_t M2,
 const size_t R1,
 const size_t R2,
 const size_t R3,
 const size_t R4,
 bool positive)
```

**13.234.3.4 PermApplyS\_double()**

```
void PermApplyS_double (
 double * A,
 const size_t lda,
 const size_t width,
 const size_t M2,
 const size_t R1,
 const size_t R2,
 const size_t R3,
 const size_t R4)
```

**13.234.3.5 MatrixApplyT\_modular\_double()**

```
void MatrixApplyT_modular_double (
 const double p,
 double * A,
 const size_t lda,
 const size_t width,
 const size_t N2,
 const size_t R1,
 const size_t R2,
 const size_t R3,
```

```
 const size_t R4,
 bool positive)
```

#### 13.234.3.6 PermApplyT\_double()

```
void PermApplyT_double (
 double * A,
 const size_t lda,
 const size_t width,
 const size_t N2,
 const size_t R1,
 const size_t R2,
 const size_t R3,
 const size_t R4)
```

#### 13.234.3.7 composePermutationsLLM()

```
void composePermutationsLLM (
 size_t * MathP,
 const size_t * P1,
 const size_t * P2,
 const size_t R,
 const size_t N)
```

#### 13.234.3.8 composePermutationsLLL()

```
void composePermutationsLLL (
 size_t * P1,
 const size_t * P2,
 const size_t R,
 const size_t N)
```

#### 13.234.3.9 composePermutationsMLM()

```
void composePermutationsMLM (
 size_t * MathP1,
 const size_t * P2,
 const size_t R,
 const size_t N)
```

#### 13.234.3.10 cyclic\_shift\_mathPerm()

```
void cyclic_shift_mathPerm (
 size_t * P,
 const size_t s)
```

#### 13.234.3.11 cyclic\_shift\_row\_modular\_double()

```
void cyclic_shift_row_modular_double (
 const double p,
 double * A,
 size_t m,
 size_t n,
 size_t lda,
 bool positive)
```

#### 13.234.3.12 cyclic\_shift\_col\_modular\_double()

```
void cyclic_shift_col_modular_double (
 const double p,
```

```
double * A,
size_t m,
size_t n,
size_t lda,
bool positive)
```

#### 13.234.3.13 applyP\_modular\_double()

```
void applyP_modular_double (
 const double p,
 const enum FFLAS_C_SIDE Side,
 const enum FFLAS_C_TRANSPOSE Trans,
 const size_t M,
 const size_t ibeg,
 const size_t iend,
 double * A,
 const size_t lda,
 const size_t * P,
 bool positive)
```

#### 13.234.3.14 fgetrsin\_modular\_double()

```
void fgetrsin_modular_double (
 const double p,
 const enum FFLAS_C_SIDE Side,
 const size_t M,
 const size_t N,
 const size_t R,
 double * A,
 const size_t lda,
 const size_t * P,
 const size_t * Q,
 double * B,
 const size_t ldb,
 int * info,
 bool positive)
```

#### 13.234.3.15 fgetrs\_modular\_double()

```
double * fgetrs_modular_double (
 const double p,
 const enum FFLAS_C_SIDE Side,
 const size_t M,
 const size_t N,
 const size_t NRHS,
 const size_t R,
 double * A,
 const size_t lda,
 const size_t * P,
 const size_t * Q,
 double * X,
 const size_t ldx,
 const double * B,
 const size_t ldb,
 int * info,
 bool positive)
```



**13.234.3.16 fgesvin\_modular\_double()**

```
size_t fgesvin_modular_double (
 const double p,
 const enum FFLAS_C_SIDE Side,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 double * B,
 const size_t ldb,
 int * info,
 bool positive)
```

**13.234.3.17 fgesv\_modular\_double()**

```
size_t fgesv_modular_double (
 const double p,
 const enum FFLAS_C_SIDE Side,
 const size_t M,
 const size_t N,
 const size_t NRHS,
 double * A,
 const size_t lda,
 double * X,
 const size_t ldx,
 const double * B,
 const size_t ldb,
 int * info)
```

**13.234.3.18 ftrtri\_modular\_double()**

```
void ftrtri_modular_double (
 const double p,
 const enum FFLAS_C_UPLO Uplo,
 const enum FFLAS_C_DIAG Diag,
 const size_t N,
 double * A,
 const size_t lda,
 bool positive)
```

**13.234.3.19 trinv\_left\_modular\_double()**

```
void trinv_left_modular_double (
 const double p,
 const size_t N,
 const double * L,
 const size_t ldl,
 double * X,
 const size_t ldx,
 bool positive)
```

**13.234.3.20 ftrtrm\_modular\_double()**

```
void ftrtrm_modular_double (
 const double p,
 const enum FFLAS_C_DIAG diag,
 const size_t N,
 double * A,
```

```
 const size_t lda,
 bool positive)
```

### 13.234.3.21 PLUQ\_modular\_double()

```
size_t PLUQ_modular_double (
 const double p,
 const enum FFLAS_C_DIAG Diag,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 size_t * P,
 size_t * Q,
 bool positive)
```

### 13.234.3.22 LUdivine\_modular\_double()

```
size_t LUdivine_modular_double (
 const double p,
 const enum FFLAS_C_DIAG Diag,
 const enum FFLAS_C_TRANSPOSE trans,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const enum FFPACK_C_LU_TAG LuTag,
 const size_t cutoff,
 bool positive)
```

### 13.234.3.23 LUdivine\_small\_modular\_double()

```
size_t LUdivine_small_modular_double (
 const double p,
 const enum FFLAS_C_DIAG Diag,
 const enum FFLAS_C_TRANSPOSE trans,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 size_t * P,
 size_t * Q,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)
```

### 13.234.3.24 LUdivine\_gauss\_modular\_double()

```
size_t LUdivine_gauss_modular_double (
 const double p,
 const enum FFLAS_C_DIAG Diag,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 size_t * P,
 size_t * Q,
```

```
const enum FFPACK_C_LU_TAG LuTag,
bool positive)
```

#### 13.234.3.25 ColumnEchelonForm\_modular\_double()

```
size_t ColumnEchelonForm_modular_double (
 const double p,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)
```

#### 13.234.3.26 RowEchelonForm\_modular\_double()

```
size_t RowEchelonForm_modular_double (
 const double p,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)
```

#### 13.234.3.27 ColumnEchelonForm\_modular\_float()

```
size_t ColumnEchelonForm_modular_float (
 const float p,
 const size_t M,
 const size_t N,
 float * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)
```

#### 13.234.3.28 RowEchelonForm\_modular\_float()

```
size_t RowEchelonForm_modular_float (
 const float p,
 const size_t M,
 const size_t N,
 float * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)
```

**13.234.3.29 ColumnEchelonForm\_modular\_int32\_t()**

```
size_t ColumnEchelonForm_modular_int32_t (
 const int32_t p,
 const size_t M,
 const size_t N,
 int32_t * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)
```

**13.234.3.30 RowEchelonForm\_modular\_int32\_t()**

```
size_t RowEchelonForm_modular_int32_t (
 const int32_t p,
 const size_t M,
 const size_t N,
 int32_t * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)
```

**13.234.3.31 ReducedColumnEchelonForm\_modular\_double()**

```
size_t ReducedColumnEchelonForm_modular_double (
 const double p,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)
```

**13.234.3.32 ReducedRowEchelonForm\_modular\_double()**

```
size_t ReducedRowEchelonForm_modular_double (
 const double p,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)
```

**13.234.3.33 ReducedColumnEchelonForm\_modular\_float()**

```
size_t ReducedColumnEchelonForm_modular_float (
 const float p,
```

```

 const size_t M,
 const size_t N,
 float * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)

```

#### 13.234.3.34 ReducedRowEchelonForm\_modular\_float()

```

size_t ReducedRowEchelonForm_modular_float (
 const float p,
 const size_t M,
 const size_t N,
 float * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)

```

#### 13.234.3.35 ReducedColumnEchelonForm\_modular\_int32\_t()

```

size_t ReducedColumnEchelonForm_modular_int32_t (
 const int32_t p,
 const size_t M,
 const size_t N,
 int32_t * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)

```

#### 13.234.3.36 ReducedRowEchelonForm\_modular\_int32\_t()

```

size_t ReducedRowEchelonForm_modular_int32_t (
 const int32_t p,
 const size_t M,
 const size_t N,
 int32_t * A,
 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)

```

#### 13.234.3.37 ReducedRowEchelonForm2\_modular\_double()

```

size_t ReducedRowEchelonForm2_modular_double (
 const double p,
 const size_t M,
 const size_t N,
 double * A,

```

```

 const size_t lda,
 size_t * P,
 size_t * Qt,
 const bool transform,
 bool positive)

```

#### 13.234.3.38 REF\_modular\_double()

```

size_t REF_modular_double (
 const double p,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 const size_t colbeg,
 const size_t rowbeg,
 const size_t colsize,
 size_t * Qt,
 size_t * P,
 bool positive)

```

#### 13.234.3.39 Invertin\_modular\_double()

```

double * Invertin_modular_double (
 const double p,
 const size_t M,
 double * A,
 const size_t lda,
 int * nullity,
 bool positive)

```

#### 13.234.3.40 Invert\_modular\_double()

```

double * Invert_modular_double (
 const double p,
 const size_t M,
 const double * A,
 const size_t lda,
 double * X,
 const size_t ldx,
 int * nullity,
 bool positive)

```

#### 13.234.3.41 Invert2\_modular\_double()

```

double * Invert2_modular_double (
 const double p,
 const size_t M,
 double * A,
 const size_t lda,
 double * X,
 const size_t ldx,
 int * nullity,
 bool positive)

```

#### 13.234.3.42 KrylovElim\_modular\_double()

```

size_t KrylovElim_modular_double (
 const double p,

```

```
const size_t M,
const size_t N,
double * A,
const size_t lda,
size_t * P,
size_t * Q,
const size_t deg,
size_t * iterates,
size_t * inviterates,
const size_t maxit,
size_t virt,
bool positive)
```

#### 13.234.3.43 SpecRankProfile\_modular\_double()

```
size_t SpecRankProfile_modular_double (
 const double p,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 const size_t deg,
 size_t * rankProfile,
 bool positive)
```

#### 13.234.3.44 Rank\_modular\_double()

```
size_t Rank_modular_double (
 const double p,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 bool positive)
```

#### 13.234.3.45 IsSingular\_modular\_double()

```
bool IsSingular_modular_double (
 const double p,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 bool positive)
```

#### 13.234.3.46 Det\_modular\_double()

```
double Det_modular_double (
 const double p,
 const size_t N,
 double * A,
 const size_t lda,
 bool positive)
```

#### 13.234.3.47 Solve\_modular\_double()

```
double * Solve_modular_double (
 const double p,
 const size_t M,
```

```

double * A,
const size_t lda,
double * x,
const int incx,
const double * b,
const int incb,
bool positive)

```

#### 13.234.3.48 solveLB\_modular\_double()

```

void solveLB_modular_double (
 const double p,
 const enum FFLAS_C_SIDE Side,
 const size_t M,
 const size_t N,
 const size_t R,
 double * L,
 const size_t ldl,
 const size_t * Q,
 double * B,
 const size_t ldb)

```

#### 13.234.3.49 solveLB2\_modular\_double()

```

void solveLB2_modular_double (
 const double p,
 const enum FFLAS_C_SIDE Side,
 const size_t M,
 const size_t N,
 const size_t R,
 double * L,
 const size_t ldl,
 const size_t * Q,
 double * B,
 const size_t ldb,
 bool positive)

```

#### 13.234.3.50 RandomNullSpaceVector\_modular\_double()

```

void RandomNullSpaceVector_modular_double (
 const double p,
 const enum FFLAS_C_SIDE Side,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 double * X,
 const size_t incX,
 bool positive)

```

#### 13.234.3.51 NullSpaceBasis\_modular\_double()

```

size_t NullSpaceBasis_modular_double (
 const double p,
 const enum FFLAS_C_SIDE Side,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,

```



```
double ** NS,
size_t * ldn,
size_t * NSdim,
bool positive)
```

#### 13.234.3.52 RowRankProfile\_modular\_double()

```
size_t RowRankProfile_modular_double (
 const double p,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 size_t ** rkprofile,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)
```

#### 13.234.3.53 ColumnRankProfile\_modular\_double()

```
size_t ColumnRankProfile_modular_double (
 const double p,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 size_t ** rkprofile,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)
```

#### 13.234.3.54 RankProfileFromLU()

```
void RankProfileFromLU (
 const size_t * P,
 const size_t N,
 const size_t R,
 size_t * rkprofile,
 const enum FFPACK_C_LU_TAG LuTag)
```

#### 13.234.3.55 LeadingSubmatrixRankProfiles()

```
size_t LeadingSubmatrixRankProfiles (
 const size_t M,
 const size_t N,
 const size_t R,
 const size_t LSm,
 const size_t LSn,
 const size_t * P,
 const size_t * Q,
 size_t * RRP,
 size_t * CRP)
```

#### 13.234.3.56 RowRankProfileSubmatrixIndices\_modular\_double()

```
size_t RowRankProfileSubmatrixIndices_modular_double (
 const double p,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
```

```

 size_t ** rowindices,
 size_t ** colindices,
 size_t * R,
 bool positive)

```

### 13.234.3.57 ColRankProfileSubmatrixIndices\_modular\_double()

```

size_t ColRankProfileSubmatrixIndices_modular_double (
 const double p,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 size_t ** rowindices,
 size_t ** colindices,
 size_t * R,
 bool positive)

```

### 13.234.3.58 RowRankProfileSubmatrix\_modular\_double()

```

size_t RowRankProfileSubmatrix_modular_double (
 const double p,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 double ** X,
 size_t * R,
 bool positive)

```

### 13.234.3.59 ColRankProfileSubmatrix\_modular\_double()

```

size_t ColRankProfileSubmatrix_modular_double (
 const double p,
 const size_t M,
 const size_t N,
 double * A,
 const size_t lda,
 double ** X,
 size_t * R,
 bool positive)

```

### 13.234.3.60 getTriangular\_modular\_double()

```

void getTriangular_modular_double (
 const double p,
 const enum FFLAS_C_UPLO Uplo,
 const enum FFLAS_C_DIAG diag,
 const size_t M,
 const size_t N,
 const size_t R,
 const double * A,
 const size_t lda,
 double * T,
 const size_t ldt,
 const bool OnlyNonZeroVectors,
 bool positive)

```

**13.234.3.61 getTriangularin\_modular\_double()**

```
void getTriangularin_modular_double (
 const double p,
 const enum FFLAS_C_UPLO Uplo,
 const enum FFLAS_C_DIAG diag,
 const size_t M,
 const size_t N,
 const size_t R,
 double * A,
 const size_t lda,
 bool positive)
```

**13.234.3.62 getEchelonForm\_modular\_double()**

```
void getEchelonForm_modular_double (
 const double p,
 const enum FFLAS_C_UPLO Uplo,
 const enum FFLAS_C_DIAG diag,
 const size_t M,
 const size_t N,
 const size_t R,
 const size_t * P,
 const double * A,
 const size_t lda,
 double * T,
 const size_t ldt,
 const bool OnlyNonZeroVectors,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)
```

**13.234.3.63 getEchelonFormin\_modular\_double()**

```
void getEchelonFormin_modular_double (
 const double p,
 const enum FFLAS_C_UPLO Uplo,
 const enum FFLAS_C_DIAG diag,
 const size_t M,
 const size_t N,
 const size_t R,
 const size_t * P,
 double * A,
 const size_t lda,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)
```

**13.234.3.64 getEchelonTransform\_modular\_double()**

```
void getEchelonTransform_modular_double (
 const double p,
 const enum FFLAS_C_UPLO Uplo,
 const enum FFLAS_C_DIAG diag,
 const size_t M,
 const size_t N,
 const size_t R,
 const size_t * P,
 const size_t * Q,
 const double * A,
 const size_t lda,
```

```
double * T,
const size_t ldt,
const enum FFPACK_C_LU_TAG LuTag,
bool positive)
```

### 13.234.3.65 getReducedEchelonForm\_modular\_double()

```
void getReducedEchelonForm_modular_double (
 const double p,
 const enum FFLAS_C_UPLO Uplo,
 const size_t M,
 const size_t N,
 const size_t R,
 const size_t * P,
 const double * A,
 const size_t lda,
 double * T,
 const size_t ldt,
 const bool OnlyNonZeroVectors,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)
```

### 13.234.3.66 getReducedEchelonFormin\_modular\_double()

```
void getReducedEchelonFormin_modular_double (
 const double p,
 const enum FFLAS_C_UPLO Uplo,
 const size_t M,
 const size_t N,
 const size_t R,
 const size_t * P,
 double * A,
 const size_t lda,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)
```

### 13.234.3.67 getReducedEchelonTransform\_modular\_double()

```
void getReducedEchelonTransform_modular_double (
 const double p,
 const enum FFLAS_C_UPLO Uplo,
 const size_t M,
 const size_t N,
 const size_t R,
 const size_t * P,
 const size_t * Q,
 const double * A,
 const size_t lda,
 double * T,
 const size_t ldt,
 const enum FFPACK_C_LU_TAG LuTag,
 bool positive)
```

### 13.234.3.68 PLUQtoEchelonPermutation()

```
void PLUQtoEchelonPermutation (
 const size_t N,
 const size_t R,
```

```
const size_t * P,
size_t * outPerm)
```

## 13.235 ffpack\_inst.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "givaro/modular.h"
#include "givaro/modular-balanced.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include "ffpack_inst_implem.inl"
```

### Macros

- `#define __FFPACK_INST_C`
- `#define FFLAS_COMPILED`
- `#define INST_OR_DECL`
- `#define FFLAS_FIELD Givaro::ModularBalanced`
- `#define FFLAS_ELT double`
- `#define FFLAS_ELT float`
- `#define FFLAS_ELT int64_t`
- `#define FFLAS_FIELD Givaro::Modular`
- `#define FFLAS_ELT double`
- `#define FFLAS_ELT float`
- `#define FFLAS_ELT int64_t`

### 13.235.1 Macro Definition Documentation

#### 13.235.1.1 \_\_FFPACK\_INST\_C

```
#define __FFPACK_INST_C
```

#### 13.235.1.2 FFLAS\_COMPILED

```
#define FFLAS_COMPILED
```

#### 13.235.1.3 INST\_OR\_DECL

```
#define INST_OR_DECL
```

#### 13.235.1.4 FFLAS\_FIELD [1/2]

```
#define FFLAS_FIELD Givaro::ModularBalanced
```

#### 13.235.1.5 FFLAS\_ELT [1/6]

```
#define FFLAS_ELT double
```

#### 13.235.1.6 FFLAS\_ELT [2/6]

```
#define FFLAS_ELT float
```

#### 13.235.1.7 FFLAS\_ELT [3/6]

```
#define FFLAS_ELT int64_t
```

#### 13.235.1.8 FFLAS\_FIELD [2/2]

```
#define FFLAS_FIELD Givaro::Modular
```

**13.235.1.9 FFLAS\_ELT [4/6]**

```
#define FFLAS_ELT double
```

**13.235.1.10 FFLAS\_ELT [5/6]**

```
#define FFLAS_ELT float
```

**13.235.1.11 FFLAS\_ELT [6/6]**

```
#define FFLAS_ELT int64_t
```

**13.236 ffpack\_inst.h File Reference**

```
#include "givaro/modular.h"
#include "givaro/modular-balanced.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include "ffpack_inst_implem.inl"
```

**Macros**

- `#define FFLAS_COMPILED`
- `#define INST_OR_DECL <>`
- `#define FFLAS_FIELD Givaro::ModularBalanced`
- `#define FFLAS_ELT double`
- `#define FFLAS_ELT float`
- `#define FFLAS_ELT int64_t`
- `#define FFLAS_FIELD Givaro::Modular`
- `#define FFLAS_ELT double`
- `#define FFLAS_ELT float`
- `#define FFLAS_ELT int64_t`

**13.236.1 Macro Definition Documentation****13.236.1.1 FFLAS\_COMPILED**

```
#define FFLAS_COMPILED
```

**13.236.1.2 INST\_OR\_DECL**

```
#define INST_OR_DECL <>
```

**13.236.1.3 FFLAS\_FIELD [1/2]**

```
#define FFLAS_FIELD Givaro::ModularBalanced
```

**13.236.1.4 FFLAS\_ELT [1/6]**

```
#define FFLAS_ELT double
```

**13.236.1.5 FFLAS\_ELT [2/6]**

```
#define FFLAS_ELT float
```

**13.236.1.6 FFLAS\_ELT [3/6]**

```
#define FFLAS_ELT int64_t
```

**13.236.1.7 FFLAS\_FIELD [2/2]**

```
#define FFLAS_FIELD Givaro::Modular
```

**13.236.1.8 FFLAS\_ELT [4/6]**

```
#define FFLAS_ELT double
```

**13.236.1.9 FFLAS\_ELT [5/6]**

```
#define FFLAS_ELT float
```

**13.236.1.10 FFLAS\_ELT [6/6]**

```
#define FFLAS_ELT int64_t
```

**13.237 ffpack\_inst\_implem.inl File Reference****Namespaces**

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

**Functions**

- void [composePermutationsLLM](#) (size\_t \*MathP, const size\_t \*P1, const size\_t \*P2, const size\_t R, const size\_t N)  
*Computes  $P1 \times \text{Diag}(I_R, P2)$  where  $P1$  is a LAPACK and  $P2$  a LAPACK permutation and store the result in  $\text{MathP}$  as a  $\text{MathPermutation}$  format.*
- void [composePermutationsLLL](#) (size\_t \*P1, const size\_t \*P2, const size\_t R, const size\_t N)  
*Computes  $P1 \times \text{Diag}(I_R, P2)$  where  $P1$  is a LAPACK and  $P2$  a LAPACK permutation and store the result in  $P1$  as a LAPACK permutation.*
- void [composePermutationsMLM](#) (size\_t \*MathP1, const size\_t \*P2, const size\_t R, const size\_t N)  
*Computes  $\text{MathP1} \times \text{Diag}(I_R, P2)$  where  $\text{MathP1}$  is a  $\text{MathPermutation}$  and  $P2$  a LAPACK permutation and store the result in  $\text{MathP1}$  as a  $\text{MathPermutation}$  format.*
- void [cyclic\\_shift\\_mathPerm](#) (size\_t \*P, const size\_t s)
- template<typename Base\_t >  
void [cyclic\\_shift\\_row\\_col](#) (Base\_t \*A, size\_t m, size\_t n, size\_t lda)
- template [INST\\_OR\\_DECL](#) void [cyclic\\_shift\\_row](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, [FFLAS\\_ELT](#) \*A, size\_t m, size\_t n, size\_t lda)
- template [INST\\_OR\\_DECL](#) void [cyclic\\_shift\\_col](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, [FFLAS\\_ELT](#) \*A, size\_t m, size\_t n, size\_t lda)
- template [INST\\_OR\\_DECL](#) void [applyP](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const [FFLAS::FFLAS\\_TRANSPOSE](#) Trans, const size\_t M, const size\_t ibeg, const size\_t iend, [FFLAS\\_ELT](#) \*A, const size\_t lda, const size\_t \*P)
- template [INST\\_OR\\_DECL](#) void [fgetrs](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, const size\_t R, [FFLAS\\_ELT](#) \*A, const size\_t lda, const size\_t \*P, const size\_t \*Q, [FFLAS\\_ELT](#) \*B, const size\_t ldb, int \*info)
- template [INST\\_OR\\_DECL](#) [FFLAS\\_ELT](#) \* [fgetrs](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, const size\_t NRHS, const size\_t R, [FFLAS\\_ELT](#) \*A, const size\_t lda, const size\_t \*P, const size\_t \*Q, [FFLAS\\_ELT](#) \*X, const size\_t ldx, const [FFLAS\\_ELT](#) \*B, const size\_t ldb, int \*info)
- template [INST\\_OR\\_DECL](#) size\_t [fgesv](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, [FFLAS\\_ELT](#) \*A, const size\_t lda, [FFLAS\\_ELT](#) \*B, const size\_t ldb, int \*info)
- template [INST\\_OR\\_DECL](#) size\_t [fgesv](#) (const [FFLAS\\_FIELD](#)< [FFLAS\\_ELT](#) > &F, const [FFLAS::FFLAS\\_SIDE](#) Side, const size\_t M, const size\_t N, const size\_t NRHS, [FFLAS\\_ELT](#) \*A, const size\_t lda, [FFLAS\\_ELT](#) \*X, const size\_t ldx, const [FFLAS\\_ELT](#) \*B, const size\_t ldb, int \*info)

- template `INST_OR_DECL` void `fttri` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `FFLAS::FFLAS_UPLO` Uplo, const `FFLAS::FFLAS_DIAG` Diag, const size\_t N, `FFLAS_ELT` \*A, const size\_t lda, const size\_t threshold)
- template `INST_OR_DECL` void `trinv_left` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t N, const `FFLAS_ELT` \*L, const size\_t ldl, `FFLAS_ELT` \*X, const size\_t ldx)
- template `INST_OR_DECL` void `fttrm` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `FFLAS::FFLAS_SIDE` side, const `FFLAS::FFLAS_DIAG` diag, const size\_t N, `FFLAS_ELT` \*A, const size\_t lda)
- template `INST_OR_DECL` size\_t `PLUQ` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `FFLAS::FFLAS_DIAG` Diag, const size\_t M, const size\_t N, `FFLAS_ELT` \*A, const size\_t lda, size\_t \*P, size\_t \*Q)
- template `INST_OR_DECL` size\_t `LUdivine` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `FFLAS::FFLAS_DIAG` Diag, const `FFLAS::FFLAS_TRANSPOSE` trans, const size\_t M, const size\_t N, `FFLAS_ELT` \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const `FFPACK_LU_TAG` LuTag, const size\_t cutoff)
- template `INST_OR_DECL` size\_t `LUdivine_small` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `FFLAS::FFLAS_DIAG` Diag, const `FFLAS::FFLAS_TRANSPOSE` trans, const size\_t M, const size\_t N, `FFLAS_ELT` \*A, const size\_t lda, size\_t \*P, size\_t \*Q, const `FFPACK_LU_TAG` LuTag)
- template `INST_OR_DECL` size\_t `LUdivine_gauss` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `FFLAS::FFLAS_DIAG` Diag, const size\_t M, const size\_t N, `FFLAS_ELT` \*A, const size\_t lda, size\_t \*P, size\_t \*Q, const `FFPACK_LU_TAG` LuTag)
- template `INST_OR_DECL` size\_t `RowEchelonForm` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t M, const size\_t N, `FFLAS_ELT` \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const `FFPACK_LU_TAG` LuTag)
- template `INST_OR_DECL` size\_t `ReducedRowEchelonForm` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t M, const size\_t N, `FFLAS_ELT` \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const `FFPACK_LU_TAG` LuTag)
- template `INST_OR_DECL` size\_t `ColumnEchelonForm` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t M, const size\_t N, `FFLAS_ELT` \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const `FFPACK_LU_TAG` LuTag)
- template `INST_OR_DECL` size\_t `ReducedColumnEchelonForm` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t M, const size\_t N, `FFLAS_ELT` \*A, const size\_t lda, size\_t \*P, size\_t \*Qt, const bool transform, const `FFPACK_LU_TAG` LuTag)
- template `INST_OR_DECL` `FFLAS_ELT` \* `Invert` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t M, `FFLAS_ELT` \*A, const size\_t lda, int &nullity)
- template `INST_OR_DECL` `FFLAS_ELT` \* `Invert` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t M, const `FFLAS_ELT` \*A, const size\_t lda, `FFLAS_ELT` \*X, const size\_t ldx, int &nullity)
- template `INST_OR_DECL` `FFLAS_ELT` \* `Invert2` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const size\_t M, `FFLAS_ELT` \*A, const size\_t lda, `FFLAS_ELT` \*X, const size\_t ldx, int &nullity)
- template `INST_OR_DECL` std::list< Givaro::Poly1Dom< `FFLAS_FIELD`< `FFLAS_ELT` > >::Element > & `CharPoly` (const Givaro::Poly1Dom< `FFLAS_FIELD`< `FFLAS_ELT` > > &R, std::list< Givaro::Poly1Dom< `FFLAS_FIELD`< `FFLAS_ELT` > >::Element > &charp, const size\_t N, `FFLAS_ELT` \*A, const size\_t lda, `FFLAS_FIELD`< `FFLAS_ELT` >::RandIter &G, const `FFPACK_CHARPOLY_TAG` CharpTag, const size\_t degree)
- template `INST_OR_DECL` Givaro::Poly1Dom< `FFLAS_FIELD`< `FFLAS_ELT` > >::Element & `CharPoly` (const Givaro::Poly1Dom< `FFLAS_FIELD`< `FFLAS_ELT` > > &R, Givaro::Poly1Dom< `FFLAS_FIELD`< `FFLAS_ELT` > >::Element &charp, const size\_t N, `FFLAS_ELT` \*A, const size\_t lda, `FFLAS_FIELD`< `FFLAS_ELT` >::RandIter &G, const `FFPACK_CHARPOLY_TAG` CharpTag, const size\_t degree)
- template `INST_OR_DECL` Givaro::Poly1Dom< `FFLAS_FIELD`< `FFLAS_ELT` > >::Element & `CharPoly` (const Givaro::Poly1Dom< `FFLAS_FIELD`< `FFLAS_ELT` > > &R, Givaro::Poly1Dom< `FFLAS_FIELD`< `FFLAS_ELT` > >::Element &charp, const size\_t N, `FFLAS_ELT` \*A, const size\_t lda, const `FFPACK_CHARPOLY_TAG` CharpTag, const size\_t degree)
- template `INST_OR_DECL` std::vector< `FFLAS_ELT` > & `MinPoly` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, std::vector< `FFLAS_ELT` > &minP, const size\_t N, const `FFLAS_ELT` \*A, const size\_t lda, `FFLAS_FIELD`< `FFLAS_ELT` >::RandIter &G)
- template `INST_OR_DECL` std::vector< `FFLAS_ELT` > & `MinPoly` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, std::vector< `FFLAS_ELT` > &minP, const size\_t N, const `FFLAS_ELT` \*A, const size\_t lda)
- template `INST_OR_DECL` std::vector< `FFLAS_ELT` > & `MatVecMinPoly` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, std::vector< `FFLAS_ELT` > &minP, const size\_t N, const `FFLAS_ELT` \*A, const size\_t lda, const `FFLAS_ELT` \*V, const size\_t incv)



- template `INST_OR_DECL` `size_t KrylovElim` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `size_t` M, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda, `size_t` \*P, `size_t` \*Q, const `size_t` deg, `size_t` \*iterates, `size_t` \*inviterates, const `size_t` maxit, `size_t` virt)
- template `INST_OR_DECL` `size_t SpecRankProfile` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `size_t` M, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda, const `size_t` deg, `size_t` \*rankProfile)
- template `INST_OR_DECL` `size_t Rank` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `size_t` M, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda)
- template `INST_OR_DECL` `bool IsSingular` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `size_t` M, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda)
- template `INST_OR_DECL` `FFLAS_ELT & Det` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, `FFLAS_ELT` &det, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda, `size_t` \*P, `size_t` \*Q)
- template `INST_OR_DECL` `FFLAS_ELT & Det` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, `FFLAS_ELT` &det, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda, const `FFLAS::ParSeqHelper::Parallel`< `FFLAS::CuttingStrategy::Recursive`, `FFLAS::StrategyParameter::Threads` > &parH, `size_t` \*P, `size_t` \*Q)
- template `INST_OR_DECL` `FFLAS_ELT * Solve` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `size_t` M, `FFLAS_ELT` \*A, const `size_t` lda, `FFLAS_ELT` \*x, const `int` incx, const `FFLAS_ELT` \*b, const `int` incb)
- template `INST_OR_DECL` `void solveLB` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `FFLAS::FFLAS_SIDE` Side, const `size_t` M, const `size_t` N, const `size_t` R, `FFLAS_ELT` \*L, const `size_t` ldl, const `size_t` \*Q, `FFLAS_ELT` \*B, const `size_t` ldb)
- template `INST_OR_DECL` `void solveLB2` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `FFLAS::FFLAS_SIDE` Side, const `size_t` M, const `size_t` N, const `size_t` R, `FFLAS_ELT` \*L, const `size_t` ldl, const `size_t` \*Q, `FFLAS_ELT` \*B, const `size_t` ldb)
- template `INST_OR_DECL` `void RandomNullSpaceVector` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `FFLAS::FFLAS_SIDE` Side, const `size_t` M, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda, `FFLAS_ELT` \*X, const `size_t` incX)
- template `INST_OR_DECL` `size_t NullSpaceBasis` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `FFLAS::FFLAS_SIDE` Side, const `size_t` M, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda, `FFLAS_ELT` \*&NS, `size_t` &ldn, `size_t` &NSdim)
- template `INST_OR_DECL` `size_t RowRankProfile` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `size_t` M, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda, `size_t` \*rkprofile, const `FFPACK_LU_TAG` LuTag)
- template `INST_OR_DECL` `size_t ColumnRankProfile` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `size_t` M, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda, `size_t` \*rkprofile, const `FFPACK_LU_TAG` LuTag)
- `void RankProfileFromLU` (const `size_t` \*P, const `size_t` N, const `size_t` R, `size_t` \*rkprofile, const `FFPACK_LU_TAG` LuTag)

*Recovers the column/row rank profile from the permutation of an LU decomposition.*

- `size_t LeadingSubmatrixRankProfiles` (const `size_t` M, const `size_t` N, const `size_t` R, const `size_t` LSm, const `size_t` LSn, const `size_t` \*P, const `size_t` \*Q, `size_t` \*RRP, `size_t` \*CRP)

*Recovers the row and column rank profiles of any leading submatrix from the PLUQ decomposition.*

- template `INST_OR_DECL` `size_t RowRankProfileSubmatrixIndices` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `size_t` M, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda, `size_t` \*rowindices, `size_t` \*colindices, `size_t` &R)
- template `INST_OR_DECL` `size_t ColRankProfileSubmatrixIndices` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `size_t` M, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda, `size_t` \*rowindices, `size_t` \*colindices, `size_t` &R)
- template `INST_OR_DECL` `size_t RowRankProfileSubmatrix` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `size_t` M, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda, `FFLAS_ELT` \*&X, `size_t` &R)
- template `INST_OR_DECL` `size_t ColRankProfileSubmatrix` (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `size_t` M, const `size_t` N, `FFLAS_ELT` \*A, const `size_t` lda, `FFLAS_ELT` \*&X, `size_t` &R)
- template `INST_OR_DECL` `void getTriangular`< `FFLAS_FIELD`< `FFLAS_ELT` > > (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `FFLAS::FFLAS_UPLO` Uplo, const `FFLAS::FFLAS_DIAG` diag, const `size_t` M, const `size_t` N, const `size_t` R, const `FFLAS_ELT` \*A, const `size_t` lda, `FFLAS_ELT` \*T, const `size_t` ldt, const `bool` OnlyNonZeroVectors)
- template `INST_OR_DECL` `void getTriangular`< `FFLAS_FIELD`< `FFLAS_ELT` > > (const `FFLAS_FIELD`< `FFLAS_ELT` > &F, const `FFLAS::FFLAS_UPLO` Uplo, const `FFLAS::FFLAS_DIAG` diag, const `size_t` M, const `size_t` N, const `size_t` R, `FFLAS_ELT` \*A, const `size_t` lda)

- template `INST_OR_DECL` void `getEchelonForm< FFLAS_FIELD< FFLAS_ELT > >` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `FFLAS::FFLAS_UPLO` Uplo, const `FFLAS::FFLAS_DIAG` diag, const `size_t` M, const `size_t` N, const `size_t` R, const `size_t` \*P, const `FFLAS_ELT` \*A, const `size_t` lda, `FFLAS_ELT` \*T, const `size_t` ldt, const bool OnlyNonZeroVectors, const `FFPACK_LU_TAG` LuTag)
- template `INST_OR_DECL` void `getEchelonForm< FFLAS_FIELD< FFLAS_ELT > >` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `FFLAS::FFLAS_UPLO` Uplo, const `FFLAS::FFLAS_DIAG` diag, const `size_t` M, const `size_t` N, const `size_t` R, const `size_t` \*P, `FFLAS_ELT` \*A, const `size_t` lda, const `FFPACK_LU_TAG` LuTag)
- template `INST_OR_DECL` void `getEchelonTransform< FFLAS_FIELD< FFLAS_ELT > >` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `FFLAS::FFLAS_UPLO` Uplo, const `FFLAS::FFLAS_DIAG` diag, const `size_t` M, const `size_t` N, const `size_t` R, const `size_t` \*P, const `size_t` \*Q, const `FFLAS_ELT` \*A, const `size_t` lda, `FFLAS_ELT` \*T, const `size_t` ldt, const `FFPACK_LU_TAG` LuTag)
- template `INST_OR_DECL` void `getReducedEchelonForm< FFLAS_FIELD< FFLAS_ELT > >` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `FFLAS::FFLAS_UPLO` Uplo, const `size_t` M, const `size_t` N, const `size_t` R, const `size_t` \*P, const `FFLAS_ELT` \*A, const `size_t` lda, `FFLAS_ELT` \*T, const `size_t` ldt, const bool OnlyNonZeroVectors, const `FFPACK_LU_TAG` LuTag)
- template `INST_OR_DECL` void `getReducedEchelonForm< FFLAS_FIELD< FFLAS_ELT > >` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `FFLAS::FFLAS_UPLO` Uplo, const `size_t` M, const `size_t` N, const `size_t` R, const `size_t` \*P, `FFLAS_ELT` \*A, const `size_t` lda, const `FFPACK_LU_TAG` LuTag)
- template `INST_OR_DECL` void `getReducedEchelonTransform< FFLAS_FIELD< FFLAS_ELT > >` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `FFLAS::FFLAS_UPLO` Uplo, const `size_t` M, const `size_t` N, const `size_t` R, const `size_t` \*P, const `size_t` \*Q, const `FFLAS_ELT` \*A, const `size_t` lda, `FFLAS_ELT` \*T, const `size_t` ldt, const `FFPACK_LU_TAG` LuTag)
- void `PLUQtoEchelonPermutation` (const `size_t` N, const `size_t` R, const `size_t` \*P, `size_t` \*outPerm)  
*Auxiliary routine: determines the permutation that changes a PLUQ decomposition into a echelon form revealing PLUQ decomposition.*
- template `INST_OR_DECL` `FFLAS_ELT` \* `LQUPTtoInverseOfFullRankMinor` (const `FFLAS_FIELD< FFLAS_ELT >` &F, const `size_t` rank, `FFLAS_ELT` \*A\_factors, const `size_t` lda, const `size_t` \*QtPointer, `FFLAS_ELT` \*X, const `size_t` ldx)

## 13.238 blockcuts.inl File Reference

```
#include <fflas-ffpack/fflas/fflas_enum.h>
#include <math.h>
#include <cassert>
```

### Data Structures

- struct [Single](#)
- struct [Row](#)
- struct [Column](#)
- struct [Block](#)
- struct [Recursive](#)
- struct [Fixed](#)
- struct [Threads](#)
- struct [Grain](#)
- struct [TwoD](#)
- struct [TwoDAdaptive](#)
- struct [ThreeD](#)
- struct [ThreeDInPlace](#)
- struct [ThreeDAdaptive](#)
- struct [Parallel< C, P >](#)
- struct [Sequential](#)
- struct [Compose< H1, H2 >](#)

## Namespaces

- namespace [FFLAS](#)
- namespace [FFLAS::CuttingStrategy](#)
- namespace [FFLAS::StrategyParameter](#)
- namespace [FFLAS::ParSeqHelper](#)

*[ParSeqHelper](#) for both *fgemm* and *ftrsm*.*

## Macros

- `#define \_\_FFLASFFPACK\_fflas\_blockcuts\_INL`
- `#define \_\_FFLASFFPACK\_MINBLOCKCUTS ((size_t)256)`

## Typedefs

- typedef [Row](#) [RNSModulus](#)

## Functions

- `template<class Cut = CuttingStrategy::Block, class Strat = StrategyParameter::Threads>  
void BlockCuts (size_t &RBLOCKSIZE, size_t &CBLOCKSIZE, const size_t m, const size_t n, const size_t numthreads)`
- `template<> void BlockCuts< CuttingStrategy::Single, StrategyParameter::Threads > (size_t &RBLOCKSIZE, size_t &CBLOCKSIZE, const size_t m, const size_t n, const size_t numthreads)`
- `template<> void BlockCuts< CuttingStrategy::Row, StrategyParameter::Fixed > (size_t &RBLOCKSIZE, size_t &CBLOCKSIZE, const size_t m, const size_t n, const size_t numthreads)`
- `template<> void BlockCuts< CuttingStrategy::Row, StrategyParameter::Grain > (size_t &RBLOCKSIZE, size_t &CBLOCKSIZE, const size_t m, const size_t n, const size_t grainsize)`
- `template<> void BlockCuts< CuttingStrategy::Block, StrategyParameter::Grain > (size_t &RBLOCKSIZE, size_t &CBLOCKSIZE, const size_t m, const size_t n, const size_t grainsize)`
- `template<> void BlockCuts< CuttingStrategy::Column, StrategyParameter::Fixed > (size_t &RBLOCKSIZE, size_t &CBLOCKSIZE, const size_t m, const size_t n, const size_t numthreads)`
- `template<> void BlockCuts< CuttingStrategy::Column, StrategyParameter::Grain > (size_t &RBLOCKSIZE, size_t &CBLOCKSIZE, const size_t m, const size_t n, const size_t grainsize)`
- `template<> void BlockCuts< CuttingStrategy::Block, StrategyParameter::Fixed > (size_t &RBLOCKSIZE, size_t &CBLOCKSIZE, const size_t m, const size_t n, const size_t numthreads)`
- `template<> void BlockCuts< CuttingStrategy::Row, StrategyParameter::Threads > (size_t &RBLOCKSIZE, size_t &CBLOCKSIZE, const size_t m, const size_t n, const size_t numthreads)`
- `template<> void BlockCuts< CuttingStrategy::Column, StrategyParameter::Threads > (size_t &RBLOCKSIZE, size_t &CBLOCKSIZE, const size_t m, const size_t n, const size_t numthreads)`
- `template<> void BlockCuts< CuttingStrategy::Block, StrategyParameter::Threads > (size_t &RBLOCKSIZE, size_t &CBLOCKSIZE, const size_t m, const size_t n, const size_t numthreads)`
- `template<class Cut = CuttingStrategy::Block, class Param = StrategyParameter::Threads>  
void BlockCuts (size_t &rowBlockSize, size_t &colBlockSize, size_t &lastRBS, size_t &lastCBS, size_t &changeRBS, size_t &changeCBS, size_t &numRowBlock, size_t &numColBlock, size_t m, size_t n, const size_t numthreads)`

## 13.238.1 Macro Definition Documentation

### 13.238.1.1 [\\_\\_FFLASFFPACK\\_fflas\\_blockcuts\\_INL](#)

```
#define __FFLASFFPACK_fflas_blockcuts_INL
```

### 13.238.1.2 [\\_\\_FFLASFFPACK\\_MINBLOCKCUTS](#)

```
#define __FFLASFFPACK_MINBLOCKCUTS ((size_t)256)
```

## 13.239 fflas\_plevel1.h File Reference

```
#include "fflas-ffpack/paladin/parallel.h"
```

### Namespaces

- namespace [FFLAS](#)

### Functions

- template<class [Field](#) >  
void [pfzero](#) (const [Field](#) &F, size\_t m, size\_t n, typename [Field::Element\\_ptr](#) C, size\_t BS=0)
- template<class [Field](#) , class [RandIter](#) >  
void [pfrand](#) (const [Field](#) &F, [RandIter](#) &G, size\_t m, size\_t n, typename [Field::Element\\_ptr](#) C, size\_t BS=0)
- template<class [Field](#) , class [Cut](#) , class [Param](#) >  
[Field::Element](#) & [fdot](#) (const [Field](#) &F, const size\_t N, typename [Field::ConstElement\\_ptr](#) x, const size\_t incx, typename [Field::ConstElement\\_ptr](#) y, const size\_t incy, typename [Field::Element](#) &d, const [ParSeqHelper::Parallel](#)< [Cut](#), [Param](#) > par)
- template<typename [Field](#) , class [Cut](#) , class [Param](#) >  
[Field::Element](#) [fdot](#) (const [Field](#) &F, const size\_t N, typename [Field::ConstElement\\_ptr](#) X, const size\_t incX, typename [Field::ConstElement\\_ptr](#) Y, const size\_t incY, const [ParSeqHelper::Parallel](#)< [Cut](#), [Param](#) > par)

## 13.240 kaapi\_routines.inl File Reference

### Macros

- #define [\\_\\_FFLASFFPACK\\_KAAPI\\_ROUTINES\\_INL](#)

### 13.240.1 Macro Definition Documentation

#### 13.240.1.1 [\\_\\_FFLASFFPACK\\_KAAPI\\_ROUTINES\\_INL](#)

```
#define __FFLASFFPACK_KAAPI_ROUTINES_INL
```

## 13.241 parallel.h File Reference

```
#include "fflas-ffpack/config.h"
#include "fflas-ffpack/paladin/blockcuts.inl"
```

### Macros

- #define [\\_\\_FFLASFFPACK\\_SEQUENTIAL](#)
- #define [index\\_t](#) size\_t
- #define [TASK](#)(M, l) {l;}
- #define [WAIT](#)
- #define [CHECK\\_DEPENDENCIES](#)
- #define [BARRIER](#)
- #define [PAR\\_BLOCK](#)
- #define [SYNCH\\_GROUP](#)(Args...) {{Args};}
- #define [THREAD\\_INDEX](#) 0
- #define [NUM\\_THREADS](#) 1
- #define [SET\\_THREADS](#)(num\_threads) {}
- #define [MAX\\_THREADS](#) 1
- #define [READ](#)(Args...)
- #define [WRITE](#)(Args...)

- `#define READWRITE(Args...)`
- `#define CONSTREFERENCE(...)`
- `#define VALUE(...)`
- `#define BEGIN_PARALLEL_MAIN(Args...) int main(Args) {`
- `#define END_PARALLEL_MAIN(void) return 0; }`
- `#define FORBLOCK1D(iter, m, Helper, Args...)`
- `#define FOR1D(i, m, Helper, Args...)`
- `#define PARFORBLOCK1D(iter, m, Helper, Args...)`
- `#define PARFOR1D(iter, m, Helper, Args...)`
- `#define FORBLOCK2D(iter, m, n, Helper, Args...)`
- `#define FOR2D(i, j, m, n, Helper, Args...)`
- `#define PARFORBLOCK2D(iter, m, n, Helper, Args...) FORBLOCK2D(iter, m, n, Helper, Args)`
- `#define PARFOR2D(i, j, m, n, Helper, Args...) FOR2D(i, j, m, n, Helper, Args)`
- `#define COMMA ,`
- `#define MODE(...) __VA_ARGS__`
- `#define RETURNPARAM(f, P1, Args...) P1=f(Args)`
- `#define NUMARGS(...) PP_NARG(__VA_ARGS__, PP_RSEQ_N())`
- `#define PP_NARG(...) PP_ARG_N(__VA_ARGS__)`
- `#define PP_ARG_N(_1, _2, _3, _4, _5, _6, _7, _8, _9, _10, _11, _12, _13, _14, _15, _16, _17, _18, _19, _20, _21, _22, _23, _24, _25, _26, _27, _28, _29, _30, _31, _32, _33, _34, _35, _36, _37, _38, _39, _40, _41, _42, _43, _44, _45, _46, _47, _48, _49, _50, _51, _52, _53, _54, _55, _56, _57, _58, _59, _60, _61, _62, _63, N, ...) N`
- `#define PP_RSEQ_N()`
- `#define NOSPLIT() FFLAS::ParSeqHelper::Sequential()`
- `#define splitting_0() FFLAS::ParSeqHelper::Parallel<FFLAS::CuttingStrategy::Block, FFLAS::StrategyParameter::Threads>()`
- `#define splitting_1(a) FFLAS::ParSeqHelper::Parallel<FFLAS::CuttingStrategy::Block, FFLAS::StrategyParameter::Threads>(a)`
- `#define splitting_2(a, c) FFLAS::ParSeqHelper::Parallel<FFLAS::CuttingStrategy::Block, c>(a)`
- `#define splitting_3(a, b, c) FFLAS::ParSeqHelper::Parallel<b, c>(a)`
- `#define splitt(_1, _2, _3, NAME, ...) NAME`
- `#define SPLITTER(...) splitt(__VA_ARGS__, splitting_3, splitting_2, splitting_1, splitting_0)(__VA_ARGS__)`

## 13.241.1 Macro Definition Documentation

### 13.241.1.1 \_\_FFLASFFPACK\_SEQUENTIAL

```
#define __FFLASFFPACK_SEQUENTIAL
```

### 13.241.1.2 index\_t

```
#define index_t size_t
```

### 13.241.1.3 TASK

```
#define TASK(
 M,
 I) {I;}
```

### 13.241.1.4 WAIT

```
#define WAIT
```

### 13.241.1.5 CHECK\_DEPENDENCIES

```
#define CHECK_DEPENDENCIES
```

### 13.241.1.6 BARRIER

```
#define BARRIER
```

**13.241.1.7 PAR\_BLOCK**

```
#define PAR_BLOCK
```

**13.241.1.8 SYNCH\_GROUP**

```
#define SYNCH_GROUP(
 Args...) {{Args}};
```

**13.241.1.9 THREAD\_INDEX**

```
#define THREAD_INDEX 0
```

**13.241.1.10 NUM\_THREADS**

```
#define NUM_THREADS 1
```

**13.241.1.11 SET\_THREADS**

```
#define SET_THREADS(
 num_threads) {}
```

**13.241.1.12 MAX\_THREADS**

```
#define MAX_THREADS 1
```

**13.241.1.13 READ**

```
#define READ(
 Args...)
```

**13.241.1.14 WRITE**

```
#define WRITE(
 Args...)
```

**13.241.1.15 READWRITE**

```
#define READWRITE(
 Args...)
```

**13.241.1.16 CONSTREFERENCE**

```
#define CONSTREFERENCE(
 ...)
```

**13.241.1.17 VALUE**

```
#define VALUE(
 ...)
```

**13.241.1.18 BEGIN\_PARALLEL\_MAIN**

```
#define BEGIN_PARALLEL_MAIN(
 Args...) int main(Args) {
```

**13.241.1.19 END\_PARALLEL\_MAIN**

```
#define END_PARALLEL_MAIN(
 void) return 0; }
```

**13.241.1.20 FORBLOCK1D**

```
#define FORBLOCK1D(
 iter,
 m,
 Helper,
 Args...)
```

**Value:**

```
{ FFLAS::ForStrategy1D<std::remove_const<decltype(m)>::type, typename decltype(Helper)::Cut, typename
 decltype(Helper)::Param> iter(m, Helper); \
 for(iter.initialize(); !iter.isTerminated(); ++iter) \
 {Args; } }
```

**13.241.1.21 FOR1D**

```
#define FOR1D(
 i,
 m,
 Helper,
 Args...)
```

**Value:**

```
FORBLOCK1D(_internal_iterator, m, Helper, \
 for(auto i=_internal_iterator.begin(); i!=_internal_iterator.end(); ++i) \
 { Args; })
```

**13.241.1.22 PARFORBLOCK1D**

```
#define PARFORBLOCK1D(
 iter,
 m,
 Helper,
 Args...)
```

**Value:**

```
for(std::remove_const<decltype(m)>::type iter=0; iter<m; ++iter) \
{ Args; }
```

**13.241.1.23 PARFOR1D**

```
#define PARFOR1D(
 iter,
 m,
 Helper,
 Args...)
```

**Value:**

```
for(std::remove_const<decltype(m)>::type iter=0; iter<m; ++iter) \
{ Args; }
```

**13.241.1.24 FORBLOCK2D**

```
#define FORBLOCK2D(
 iter,
 m,
 n,
 Helper,
 Args...)
```

**Value:**

```
{ FFLAS::ForStrategy2D<std::remove_const<decltype(m)>::type, typename decltype(Helper)::Cut, typename
 decltype(Helper)::Param> iter(m,n,Helper); \
 for(iter.initialize(); !iter.isTerminated(); ++iter) \
 { Args; } }
```

**13.241.1.25 FOR2D**

```
#define FOR2D(
 i,
```

```

 j,
 m,
 n,
 Helper,
 Args...)

```

**Value:**

```

FORBLOCK2D(_internal_iterator, m, n, Helper,
 for(auto i=_internal_iterator.ibegin(); i!=_internal_iterator.iend(); ++i) \
 for(auto j=_internal_iterator.jbegin(); j!=_internal_iterator.jend(); ++j) \
 { Args; })

```

**13.241.1.26 PARFORBLOCK2D**

```

#define PARFORBLOCK2D(
 iter,
 m,
 n,
 Helper,
 Args...) FORBLOCK2D(iter, m, n, Helper, Args)

```

**13.241.1.27 PARFOR2D**

```

#define PARFOR2D(
 i,
 j,
 m,
 n,
 Helper,
 Args...) FOR2D(i, j, m, n, Helper, Args)

```

**13.241.1.28 COMMA**

```

#define COMMA ,

```

**13.241.1.29 MODE**

```

#define MODE(
 ...) __VA_ARGS__

```

**13.241.1.30 RETURNPARAM**

```

#define RETURNPARAM(
 f,
 Pl,
 Args...) Pl=f(Args)

```

**13.241.1.31 NUMARGS**

```

#define NUMARGS(
 ...) PP_NARG__(__VA_ARGS__, PP_RSEQ_N())

```

**13.241.1.32 PP\_NARG\_**

```

#define PP_NARG_(
 ...) PP_ARG_N(__VA_ARGS__)

```

**13.241.1.33 PP\_ARG\_N**

```

#define PP_ARG_N(
 _1,
 _2,

```



\_3,  
\_4,  
\_5,  
\_6,  
\_7,  
\_8,  
\_9,  
\_10,  
\_11,  
\_12,  
\_13,  
\_14,  
\_15,  
\_16,  
\_17,  
\_18,  
\_19,  
\_20,  
\_21,  
\_22,  
\_23,  
\_24,  
\_25,  
\_26,  
\_27,  
\_28,  
\_29,  
\_30,  
\_31,  
\_32,  
\_33,  
\_34,  
\_35,  
\_36,  
\_37,  
\_38,  
\_39,  
\_40,  
\_41,  
\_42,  
\_43,  
\_44,  
\_45,  
\_46,  
\_47,  
\_48,  
\_49,  
\_50,  
\_51,  
\_52,  
\_53,  
\_54,  
\_55,  
\_56,  
\_57,  
\_58,  
\_59,  
\_60,

```

 _61,
 _62,
 _63,
 N,
 ...) N

```

### 13.241.1.34 PP\_RSEQ\_N

```
#define PP_RSEQ_N()
```

#### Value:

```

63, 62, 61, 60, \
59, 58, 57, 56, 55, 54, 53, 52, 51, 50, \
49, 48, 47, 46, 45, 44, 43, 42, 41, 40, \
39, 38, 37, 36, 35, 34, 33, 32, 31, 30, \
29, 28, 27, 26, 25, 24, 23, 22, 21, 20, \
19, 18, 17, 16, 15, 14, 13, 12, 11, 10, \
9, 8, 7, 6, 5, 4, 3, 2, 1, 0

```

### 13.241.1.35 NOSPLIT

```
#define NOSPLIT() FFLAS::ParSeqHelper::Sequential()
```

### 13.241.1.36 splitting\_0

```
#define splitting_0() FFLAS::ParSeqHelper::Parallel<FFLAS::CuttingStrategy::Block, FFLAS::StrategyParameter::Thread>
```

### 13.241.1.37 splitting\_1

```

#define splitting_1(
 a) FFLAS::ParSeqHelper::Parallel<FFLAS::CuttingStrategy::Block, FFLAS::StrategyParameter::Thread>

```

### 13.241.1.38 splitting\_2

```

#define splitting_2(
 a,
 c) FFLAS::ParSeqHelper::Parallel<FFLAS::CuttingStrategy::Block, c>(a)

```

### 13.241.1.39 splitting\_3

```

#define splitting_3(
 a,
 b,
 c) FFLAS::ParSeqHelper::Parallel<b, c>(a)

```

### 13.241.1.40 splitt

```

#define splitt(
 _1,
 _2,
 _3,
 NAME,
 ...) NAME

```

### 13.241.1.41 SPLITTER

```

#define SPLITTER(
 ...) splitt(__VA_ARGS__, splitting_3, splitting_2, splitting_1, splitting_0) (↔
__VA_ARGS__)

```

## 13.242 pfgemm\_variants.inl File Reference

### Namespaces

- namespace [FFLAS](#)

### Functions

- `template<class Field , class AlgoT , class FieldTrait >`  
`Field::Element * pfgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb,`  
`const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, const typename`  
`Field::ConstElement_ptr A, const size_t lda, const typename Field::ConstElement_ptr B, const size_t ldb,`  
`const typename Field::Element beta, typename Field::Element *C, const size_t ldc, MMHelper< Field, Al-`  
`goT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Block, StrategyParameter::Threads > > &H)`
- `template<class Field , class AlgoT , class FieldTrait >`  
`Field::Element * pfgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb,`  
`const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, const typename`  
`Field::ConstElement_ptr AA, const size_t lda, const typename Field::ConstElement_ptr BB, const size_t`  
`ldb, const typename Field::Element beta, typename Field::Element *C, const size_t ldc, MMHelper< Field,`  
`AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Recursive, StrategyParameter::ThreeDAdaptive`  
`> > &H)`
- `template<class Field , class AlgoT , class FieldTrait >`  
`Field::Element * pfgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb,`  
`const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, const typename`  
`Field::ConstElement_ptr AA, const size_t lda, const typename Field::ConstElement_ptr BB, const size_t`  
`ldb, const typename Field::Element beta, typename Field::Element *C, const size_t ldc, MMHelper< Field,`  
`AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Recursive, StrategyParameter::TwoDAdaptive >`  
`> &H)`
- `template<class Field , class AlgoT , class FieldTrait >`  
`Field::Element * pfgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb,`  
`const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, const typename`  
`Field::ConstElement_ptr AA, const size_t lda, const typename Field::ConstElement_ptr BB, const size_t`  
`ldb, const typename Field::Element beta, typename Field::Element *C, const size_t ldc, MMHelper< Field,`  
`AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Recursive, StrategyParameter::TwoD > > &H)`
- `template<class Field , class AlgoT , class FieldTrait >`  
`Field::Element_ptr pfgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE`  
`tb, const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, const typename`  
`Field::ConstElement_ptr A, const size_t lda, const typename Field::ConstElement_ptr B, const size_t ldb,`  
`const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field,`  
`AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Recursive, StrategyParameter::ThreeD > > &H)`
- `template<class Field , class AlgoT , class FieldTrait >`  
`Field::Element * pfgemm (const Field &F, const FFLAS_TRANSPOSE ta, const FFLAS_TRANSPOSE tb,`  
`const size_t m, const size_t n, const size_t k, const typename Field::Element alpha, const typename`  
`Field::ConstElement_ptr A, const size_t lda, const typename Field::ConstElement_ptr B, const size_t ldb,`  
`const typename Field::Element beta, typename Field::Element_ptr C, const size_t ldc, MMHelper< Field, Al-`  
`goT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Recursive, StrategyParameter::ThreeDInPlace >`  
`> &H)`

## 13.243 pfgemv.inl File Reference

### Namespaces

- namespace [FFLAS](#)

### Functions

- `template<class Field , class AlgoT , class FieldTrait >`  
`Field::Element_ptr fgemv (const Field &F, const FFLAS_TRANSPOSE ta, const size_t m, const size_t`  
`n, const typename Field::Element alpha, const typename Field::ConstElement_ptr A, const size_t lda,`

```
const typename Field::ConstElement_ptr X, const size_t incX, const typename Field::Element beta, type-
name Field::Element_ptr Y, const size_t incY, MMHelper< Field, AlgoT, FieldTrait, ParSeqHelper::Parallel<
CuttingStrategy::Recursive, StrategyParameter::Threads > > &H)
```

- template<class Field, class AlgoT, class FieldTrait, class Cut >  
Field::Element\_ptr fgemv (const Field &F, const FFLAS\_TRANSPOSE ta, const size\_t m, const size\_t n, const typename Field::Element alpha, const typename Field::ConstElement\_ptr A, const size\_t lda, const typename Field::ConstElement\_ptr X, const size\_t incX, const typename Field::Element beta, type-  
name Field::Element\_ptr Y, const size\_t incY, MMHelper< Field, AlgoT, FieldTrait, ParSeqHelper::Parallel< CuttingStrategy::Row, Cut > > &H)

## 13.244 align-allocator.h File Reference

```
#include "fflas-ffpack/config.h"
```

## 13.245 args-parser.h File Reference

```
#include <fflas-ffpack/fflas-ffpack-config.h>
#include <givaro/givinteger.h>
#include <givaro/givprint.h>
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <cstring>
#include <list>
#include <stdlib.h>
```

### Data Structures

- struct [Argument](#)

### Namespaces

- namespace [FFLAS](#)

### Macros

- #define [TYPE\\_BOOL](#) [TYPE\\_NONE](#)
- #define [END\\_OF\\_ARGUMENTS](#) { '\0', "\0", "\0", [TYPE\\_NONE](#), NULL }
- #define [type\\_integer](#) long int

### Enumerations

- enum [ArgumentType](#) {  
  [TYPE\\_NONE](#), [TYPE\\_INT](#), [TYPE\\_UINT64](#), [TYPE\\_LONGLONG](#),  
  [TYPE\\_INTEGER](#), [TYPE\\_DOUBLE](#), [TYPE\\_INTLIST](#), [TYPE\\_STR](#) }

### Functions

- void [parseArguments](#) (int argc, char \*\*argv, [Argument](#) \*args, bool printDefaults=true)
- void [printHelpMessage](#) (const char \*program, [Argument](#) \*args, bool printDefaults=false)
- [Argument](#) \* [findArgument](#) ([Argument](#) \*args, char c)
- int [getListArgs](#) (std::list< int > &outlist, std::string &instring)  
  *transforms a string list of ints to a list of int string "12,13,15" is turned into list of ints {12,13,15}*
- char \* [getArgumentValue](#) (int argc, char \*\*argv, int i)

*Get the value of an argument and avoid core dump when no value was given after an argument.*

- `std::ostream & writeCommandString (std::ostream &os, Argument *args, const char *programName=nullptr)`  
*writes the values of all arguments, preceded by the programName*

## 13.245.1 Macro Definition Documentation

### 13.245.1.1 TYPE\_BOOL

```
#define TYPE_BOOL TYPE_NONE
```

### 13.245.1.2 END\_OF\_ARGUMENTS

```
#define END_OF_ARGUMENTS { '\0', "\0", "\0", TYPE_NONE, NULL }
```

### 13.245.1.3 type\_integer

```
#define type_integer long int
```

## 13.245.2 Enumeration Type Documentation

### 13.245.2.1 ArgumentType

```
enum ArgumentType
```

Enumerator

|               |  |
|---------------|--|
| TYPE_NONE     |  |
| TYPE_INT      |  |
| TYPE_UINT64   |  |
| TYPE_LONGLONG |  |
| TYPE_INTEGER  |  |
| TYPE_DOUBLE   |  |
| TYPE_INTLIST  |  |
| TYPE_STR      |  |

## 13.245.3 Function Documentation

### 13.245.3.1 printHelpMessage()

```
void printHelpMessage (
 const char * program,
 Argument * args,
 bool printDefaults = false)
```

### 13.245.3.2 findArgument()

```
Argument * findArgument (
 Argument * args,
 char c)
```

### 13.245.3.3 getListArgs()

```
int getListArgs (
 std::list< int > & outlist,
 std::string & instring)
```

transforms a string list of ints to a list of int string "12,13,15" is turned into list of ints {12,13,15}

## Parameters

|                 |                      |
|-----------------|----------------------|
| <i>outlist</i>  | list once converted  |
| <i>instring</i> | list to be converted |

## Returns

status message.

## 13.246 bit\_manipulation.h File Reference

```
#include <givaro/udl.h>
#include "fflas-ffpack/fflas-ffpack-config.h"
```

## Macros

- `#define __has_builtin(x) 0`

## Functions

- `int32_t clz (uint64_t val)`
- `int32_t clz (uint32_t val)`
- `int32_t ctz (uint32_t val)`
- `int32_t ctz (uint64_t val)`

### 13.246.1 Macro Definition Documentation

#### 13.246.1.1 \_\_has\_builtin

```
#define __has_builtin(
 x) 0
```

### 13.246.2 Function Documentation

#### 13.246.2.1 clz() [1/2]

```
int32_t clz (
 uint64_t val) [inline]
```

#### 13.246.2.2 clz() [2/2]

```
int32_t clz (
 uint32_t val) [inline]
```

#### 13.246.2.3 ctz() [1/2]

```
int32_t ctz (
 uint32_t val) [inline]
```

#### 13.246.2.4 ctz() [2/2]

```
int32_t ctz (
 uint64_t val) [inline]
```

## 13.247 cast.h File Reference

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

### Functions

- template<class T, class CT = const T>  
T [fflas\\_const\\_cast](#) (CT x)

## 13.248 debug.h File Reference

Various utilities for debugging.

```
#include <fflas-ffpack/fflas-ffpack-config.h>
#include <iostream>
#include <sstream>
#include <cmath>
#include <stdexcept>
```

### Data Structures

- class [Failure](#)  
*A precondition failed.*

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

### Macros

- #define [FFLASFFPACK\\_check](#)(check)
- #define [FFLASFFPACK\\_abort](#)(msg)

### Functions

- [Failure](#) & [failure](#) ()
- template<class T >  
bool [isOdd](#) (const T &a)
- bool [isOdd](#) (const float &a)
- bool [isOdd](#) (const double &a)

### 13.248.1 Detailed Description

Various utilities for debugging.

**Todo** we should put vector printing elsewhere.

### 13.248.2 Macro Definition Documentation

#### 13.248.2.1 FFLASFFPACK\_check

```
#define FFLASFFPACK_check(
 check)
```

**Value:**

```
if (!(check)) {\
```

```

FFPACK::failure() (__func__, __FILE__, __LINE__, #check); \
throw std::runtime_error(#check); \
}

```

### 13.248.2.2 FFLASFFPACK\_abort

```

#define FFLASFFPACK_abort(
 msg)

```

#### Value:

```

{ \
 FFPACK::failure() (__func__, __FILE__, __LINE__, msg); \
 throw std::runtime_error(msg); \
}

```

## 13.249 fflas\_intrinsic.h File Reference

## 13.250 fflas\_io.h File Reference

```

#include <cstring>
#include <stdio.h>
#include <stdlib.h>
#include <fstream>
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas_memory.h"

```

### Namespaces

- namespace [FFLAS](#)

### Enumerations

- enum [FFLAS\\_FORMAT](#) {  
[FflasAuto](#) = 0 , [FflasDense](#) = 1 , [FflasSMS](#) = 2 , [FflasBinary](#) = 3 ,  
[FflasMath](#) = 4 , [FflasMaple](#) = 5 , [FflasSageMath](#) = 6 }

### Functions

- template<class [Field](#) >  
std::ostream & [WriteMatrix](#) (std::ostream &c, const [Field](#) &F, size\_t m, size\_t n, typename [Field::ConstElement\\_ptr](#) A, size\_t lda, [FFLAS\\_FORMAT](#) format, bool column\_major)  
*WriteMatrix: write a matrix to an output stream.*
- void [preamble](#) (std::ifstream &ifs, [FFLAS\\_FORMAT](#) &format)
- template<class [Field](#) >  
[Field::Element\\_ptr](#) [ReadMatrix](#) (std::ifstream &ifs, [Field](#) &F, size\_t &m, size\_t &n, typename [Field::Element\\_ptr](#) &A, [FFLAS\\_FORMAT](#) format=[FflasAuto](#))  
*ReadMatrix: read a matrix from an input stream.*
- template<class [Field](#) >  
[Field::Element\\_ptr](#) [ReadMatrix](#) (const std::string &matrix\_file, [Field](#) &F, size\_t &m, size\_t &n, typename [Field::Element\\_ptr](#) &A, [FFLAS\\_FORMAT](#) format=[FflasAuto](#))  
*ReadMatrix: read a matrix from a file.*
- template<class [Field](#) >  
void [WriteMatrix](#) (std::string &matrix\_file, const [Field](#) &F, int m, int n, typename [Field::ConstElement\\_ptr](#) A, size\_t lda, [FFLAS\\_FORMAT](#) format=[FflasDense](#), bool column\_major=false)  
*WriteMatrix: write a matrix to a file.*
- std::ostream & [WritePermutation](#) (std::ostream &c, const size\_t \*P, size\_t N)  
*WritePermutation: write a permutation matrix to an output stream.*



## 13.251 fflas\_memory.h File Reference

```
#include "fflas-ffpack/utils/align-allocator.h"
#include <givaro/givinteger.h>
```

### Namespaces

- namespace [FFLAS](#)

### Functions

- template<class Element >  
bool [alignable](#) ()
- template<> bool [alignable](#)< Givaro::Integer \* > ()
- template<class Field >  
[Field::Element\\_ptr fflas\\_new](#) (const [Field](#) &F, const size\_t m, const Alignment align=Alignment::DEFAULT)
- template<class Field >  
[Field::Element\\_ptr fflas\\_new](#) (const [Field](#) &F, const size\_t m, const size\_t n, const Alignment align=Alignment::DEFAULT)
- template<class Element >  
Element \* [fflas\\_new](#) (const size\_t m, const Alignment align=Alignment::DEFAULT)
- template<class Element\_ptr >  
void [fflas\\_delete](#) (Element\_ptr A)
- template<class Ptr, class ... Args>  
void [fflas\\_delete](#) (Ptr p, Args ... args)
- void [prefetch](#) (const int64\_t \*)
- void [getTLBSize](#) (int &tlb)
- void [queryCacheSizes](#) (int &l1, int &l2, int &l3)
- int [queryL1CacheSize](#) ()
- int [queryTopLevelCacheSize](#) ()

## 13.252 fflas\_randommatrix.h File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/utils/debug.h"
#include "fflas-ffpack/fflas/fflas.h"
#include <givaro/givinteger.h>
#include <givaro/givintprime.h>
#include <givaro/givranditer.h>
#include "fflas-ffpack/ffpack/ffpack.h"
```

### Namespaces

- namespace [FFPACK](#)  
*Finite Field **PACK** Set of elimination based routines for dense linear algebra.*

### Functions

- template<class Field, class RandIter >  
[Field::Element\\_ptr NonZeroRandomMatrix](#) (const [Field](#) &F, size\_t m, size\_t n, typename [Field::Element\\_ptr](#) A, size\_t lda, RandIter &G)  
*Random non-zero Matrix.*
- template<class Field, class RandIter >  
[Field::Element\\_ptr NonZeroRandomMatrix](#) (const [Field](#) &F, size\_t m, size\_t n, typename [Field::Element\\_ptr](#) A, size\_t lda)

*Random non-zero Matrix.*

- template<class [Field](#) , class [Randlter](#) >  
[Field::Element\\_ptr](#) [RandomMatrix](#) (const [Field](#) &F, size\_t m, size\_t n, typename [Field::Element\\_ptr](#) A, size\_t lda, [Randlter](#) &G)

*Random Matrix.*

- template<class [Field](#) >  
[Field::Element\\_ptr](#) [RandomMatrix](#) (const [Field](#) &F, size\_t m, size\_t n, typename [Field::Element\\_ptr](#) A, size\_t lda)

*Random Matrix.*

- template<class [Field](#) , class [Randlter](#) >  
[Field::Element\\_ptr](#) [RandomTriangularMatrix](#) (const [Field](#) &F, size\_t m, size\_t n, const [FFLAS::FFLAS\\_UPLO](#) UpLo, const [FFLAS::FFLAS\\_DIAG](#) Diag, bool nonsingular, typename [Field::Element\\_ptr](#) A, size\_t lda, [Randlter](#) &G)

*Random Triangular Matrix.*

- template<class [Field](#) >  
[Field::Element\\_ptr](#) [RandomTriangularMatrix](#) (const [Field](#) &F, size\_t m, size\_t n, const [FFLAS::FFLAS\\_UPLO](#) UpLo, const [FFLAS::FFLAS\\_DIAG](#) Diag, bool nonsingular, typename [Field::Element\\_ptr](#) A, size\_t lda)

*Random Triangular Matrix.*

- size\_t [RandInt](#) (size\_t a, size\_t b)
- template<class [Field](#) , class [Randlter](#) >  
[Field::Element\\_ptr](#) [RandomSymmetricMatrix](#) (const [Field](#) &F, size\_t n, bool nonsingular, typename [Field::Element\\_ptr](#) A, size\_t lda, [Randlter](#) &G)

*Random Symmetric Matrix.*

- template<class [Field](#) , class [Randlter](#) >  
[Field::Element\\_ptr](#) [RandomMatrixWithRank](#) (const [Field](#) &F, size\_t m, size\_t n, size\_t r, typename [Field::Element\\_ptr](#) A, size\_t lda, [Randlter](#) &G)

*Random Matrix with prescribed rank.*

- template<class [Field](#) >  
[Field::Element\\_ptr](#) [RandomMatrixWithRank](#) (const [Field](#) &F, size\_t m, size\_t n, size\_t r, typename [Field::Element\\_ptr](#) A, size\_t lda)

*Random Matrix with prescribed rank.*

- size\_t \* [RandomIndexSubset](#) (size\_t N, size\_t R, size\_t \*P)  
*Pick uniformly at random a sequence of  $R$  distinct elements from the set  $\{0, \dots, N - 1\}$  using Knuth's shuffle.*
- size\_t \* [RandomPermutation](#) (size\_t N, size\_t \*P)  
*Pick uniformly at random a permutation of size  $N$  stored in LAPACK format using Knuth's shuffle.*
- void [RandomRankProfileMatrix](#) (size\_t M, size\_t N, size\_t R, size\_t \*rows, size\_t \*cols)  
*Pick uniformly at random an  $R$ -subpermutation of dimension  $M \times N$  : a matrix with only  $R$  non-zeros equal to one, in a random rook placement.*

- void [swapval](#) (size\_t k, size\_t N, size\_t \*P, size\_t val)
- void [RandomSymmetricRankProfileMatrix](#) (size\_t N, size\_t R, size\_t \*rows, size\_t \*cols)  
*Pick uniformly at random a symmetric  $R$ -subpermutation of dimension  $N \times N$  : a symmetric matrix with only  $R$  non-zeros, all equal to one, in a random rook placement.*

- void [RandomLTQSRankProfileMatrix](#) (size\_t n, size\_t r, size\_t t, size\_t \*rows, size\_t \*cols)
- template<class [Field](#) , class [Randlter](#) >  
[Field::Element\\_ptr](#) [RandomMatrixWithRankandRPM](#) (const [Field](#) &F, size\_t M, size\_t N, size\_t R, typename [Field::Element\\_ptr](#) A, size\_t lda, const size\_t \*RRP, const size\_t \*CRP, [Randlter](#) &G)

*Random Matrix with prescribed rank and rank profile matrix Creates an  $m \times n$  matrix with random entries and rank  $r$ .*

- template<class [Field](#) >  
[Field::Element\\_ptr](#) [RandomMatrixWithRankandRPM](#) (const [Field](#) &F, size\_t M, size\_t N, size\_t R, typename [Field::Element\\_ptr](#) A, size\_t lda, const size\_t \*RRP, const size\_t \*CRP)

*Random Matrix with prescribed rank and rank profile matrix Creates an  $m \times n$  matrix with random entries and rank  $r$ .*

- template<class [Field](#) , class [Randlter](#) >  
[Field::Element\\_ptr](#) [RandomSymmetricMatrixWithRankandRPM](#) (const [Field](#) &F, size\_t N, size\_t R, typename [Field::Element\\_ptr](#) A, size\_t lda, const size\_t \*RRP, const size\_t \*CRP, [Randlter](#) &G)

- Random Symmetric Matrix with prescribed rank and rank profile matrix* Creates an  $n \times n$  symmetric matrix with random entries and rank  $r$ .
- `template<class Field >`  
`Field::Element_ptr RandomSymmetricMatrixWithRankandRPM` (const `Field` &F, size\_t M, size\_t N, size\_t R, typename `Field::Element_ptr` A, size\_t lda, const size\_t \*RRP, const size\_t \*CRP)  
*Random Symmetric Matrix with prescribed rank and rank profile matrix* Creates an  $n \times n$  symmetric matrix with random entries and rank  $r$ .
  - `template<class Field , class RandIter >`  
`Field::Element_ptr RandomMatrixWithRankandRandomRPM` (const `Field` &F, size\_t M, size\_t N, size\_t R, typename `Field::Element_ptr` A, size\_t lda, RandIter &G)  
*Random Matrix with prescribed rank, with random rank profile matrix* Creates an  $m \times n$  matrix with random entries, rank  $r$  and with a rank profile matrix chosen uniformly at random.
  - `template<class Field >`  
`Field::Element_ptr RandomMatrixWithRankandRandomRPM` (const `Field` &F, size\_t M, size\_t N, size\_t R, typename `Field::Element_ptr` A, size\_t lda)  
*Random Matrix with prescribed rank, with random rank profile matrix* Creates an  $m \times n$  matrix with random entries, rank  $r$  and with a rank profile matrix chosen uniformly at random.
  - `template<class Field , class RandIter >`  
`Field::Element_ptr RandomSymmetricMatrixWithRankandRandomRPM` (const `Field` &F, size\_t N, size\_t R, typename `Field::Element_ptr` A, size\_t lda, RandIter &G)  
*Random Symmetric Matrix with prescribed rank, with random rank profile matrix* Creates an  $n \times n$  matrix with random entries, rank  $r$  and with a rank profile matrix chosen uniformly at random.
  - `template<class Field >`  
`Field::Element_ptr RandomSymmetricMatrixWithRankandRandomRPM` (const `Field` &F, size\_t N, size\_t R, typename `Field::Element_ptr` A, size\_t lda)  
*Random Symmetric Matrix with prescribed rank, with random rank profile matrix* Creates an  $n \times n$  matrix with random entries, rank  $r$  and with a rank profile matrix chosen uniformly at random.
  - `template<class Field >`  
`Field::Element_ptr RandomMatrixWithDet` (const `Field` &F, size\_t n, const typename `Field::Element` d, typename `Field::Element_ptr` A, size\_t lda)  
*Random Matrix with prescribed det.*
  - `template<class Field , class RandIter >`  
`Field::Element_ptr RandomMatrixWithDet` (const `Field` &F, size\_t n, const typename `Field::Element` d, typename `Field::Element_ptr` A, size\_t lda, RandIter &G)  
*Random Matrix with prescribed det.*
  - `template<class Field , class RandIter >`  
`Field::Element_ptr RandomLTQSMMatrixWithRankandQSorder` (`Field` &F, size\_t n, size\_t r, size\_t t, typename `Field::Element_ptr` A, size\_t lda, RandIter &G)

## 13.253 flimits.h File Reference

```
#include <climits>
#include <limits>
#include <type_traits>
#include <givaro/givinteger.h>
```

### Data Structures

- `struct limits< unsigned char >`
- `struct limits< signed char >`
- `struct limits< char >`
- `struct limits< unsigned short int >`
- `struct limits< short int >`
- `struct limits< unsigned int >`
- `struct limits< int >`

- struct [limits< unsigned long >](#)
- struct [limits< long >](#)
- struct [limits< unsigned long long >](#)
- struct [limits< long long >](#)
- struct [limits< float >](#)
- struct [limits< double >](#)
- struct [limits< Givaro::Integer >](#)
- struct [limits< Reclnt::ruint< K > >](#)
- struct [limits< Reclnt::rint< K > >](#)

## Functions

- template<class T , class E >  
std::enable\_if< std::is\_signed< T >::value==std::is\_signed< E >::value, bool >::type [in\\_range](#) (E e)
- template<class T , class E >  
std::enable\_if<(std::is\_signed< T >::value)&&!(std::is\_signed< E >::value), bool >::type [in\\_range](#) (E e)
- template<class T , class E >  
std::enable\_if<!(std::is\_signed< T >::value)&&(std::is\_signed< E >::value), bool >::type [in\\_range](#) (E e)

## 13.253.1 Function Documentation

### 13.253.1.1 in\_range() [1/3]

```
template<class T , class E >
std::enable_if< std::is_signed< T >::value==std::is_signed< E >::value, bool >::type in_↵
range (
 E e)
```

### 13.253.1.2 in\_range() [2/3]

```
template<class T , class E >
std::enable_if<(std::is_signed< T >::value)&&!(std::is_signed< E >::value), bool >::type
in_range (
 E e)
```

### 13.253.1.3 in\_range() [3/3]

```
template<class T , class E >
std::enable_if<!(std::is_signed< T >::value)&&(std::is_signed< E >::value), bool >::type
in_range (
 E e)
```

## 13.254 Matio.h File Reference

```
#include <cstring>
#include <stdio.h>
#include <stdlib.h>
#include "fflas_memory.h"
```

## Functions

- template<class [Field](#) >  
[Field::Element\\_ptr read\\_field](#) (const [Field](#) &F, const char \*mat\_file, size\_t \*tni, size\_t \*tnj)
- template<class [Field](#) >  
std::ostream & [write\\_field](#) (const [Field](#) &F, std::ostream &c, typename [Field::ConstElement\\_ptr](#) E, int n, int m, int id, bool mapleFormat=false, bool column\_major=false)

## 13.254.1 Function Documentation

### 13.254.1.1 read\_field()

```
template<class Field >
Field::Element_ptr read_field (
 const Field & F,
 const char * mat_file,
 size_t * tni,
 size_t * tnj)
```

### 13.254.1.2 write\_field()

```
template<class Field >
std::ostream & write_field (
 const Field & F,
 std::ostream & c,
 typename Field::ConstElement_ptr E,
 int n,
 int m,
 int id,
 bool mapleFormat = false,
 bool column_major = false)
```

## 13.255 test-utils.h File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/utils/debug.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include "fflas-ffpack/utils/fflas_randommatrix.h"
#include <givaro/givinteger.h>
#include <givaro/givintprime.h>
#include <givaro/givranditer.h>
#include <givaro/givtimer.h>
#include <random>
#include <functional>
```

### Namespaces

- namespace [FFLAS](#)
- namespace [FFPACK](#)

*Finite Field PACK Set of elimination based routines for dense linear algebra.*

### Functions

- uint64\_t [getSeed](#) ()
- template<typename Field >  
Field \* [chooseField](#) (Givaro::Integer q, uint64\_t b, uint64\_t seed)
- template<> Givaro::ZRing< int32\_t > \* [chooseField](#)< Givaro::ZRing< int32\_t > > (Givaro::Integer q, uint64\_t b, uint64\_t seed)
- template<> Givaro::ZRing< int64\_t > \* [chooseField](#)< Givaro::ZRing< int64\_t > > (Givaro::Integer q, uint64\_t b, uint64\_t seed)
- template<> Givaro::ZRing< float > \* [chooseField](#)< Givaro::ZRing< float > > (Givaro::Integer q, uint64\_t b, uint64\_t seed)
- template<> Givaro::ZRing< double > \* [chooseField](#)< Givaro::ZRing< double > > (Givaro::Integer q, uint64\_t b, uint64\_t seed)

## 13.256 timer.h File Reference

```
#include <time.h>
#include <givaro/givtimer.h>
```

### Namespaces

- namespace [FFLAS](#)

### Typedefs

- typedef Givaro::Timer [Timer](#)
- typedef Givaro::BaseTimer [BaseTimer](#)
- typedef Givaro::UserTimer [UserTimer](#)
- typedef Givaro::SysTimer [SysTimer](#)

## 13.257 cblas.C File Reference

```
#include "fflas-ffpack/config-blas.h"
```

### Macros

- #define [\\_\\_FFLASFFPACK\\_CONFIGURATION](#)
- #define [\\_\\_FFLASFFPACK\\_HAVE\\_CBLAS](#) 1

### Functions

- int [main](#) ()

### 13.257.1 Macro Definition Documentation

#### 13.257.1.1 [\\_\\_FFLASFFPACK\\_CONFIGURATION](#)

```
#define __FFLASFFPACK_CONFIGURATION
```

#### 13.257.1.2 [\\_\\_FFLASFFPACK\\_HAVE\\_CBLAS](#)

```
#define __FFLASFFPACK_HAVE_CBLAS 1
```

### 13.257.2 Function Documentation

#### 13.257.2.1 [main\(\)](#)

```
int main (
 void)
```

## 13.258 clapack.C File Reference

```
#include "fflas-ffpack/config-blas.h"
```

### Macros

- #define [\\_\\_FFLASFFPACK\\_CONFIGURATION](#)
- #define [\\_\\_FFLASFFPACK\\_HAVE\\_LAPACK](#) 1
- #define [\\_\\_FFLASFFPACK\\_HAVE\\_CLAPACK](#) 1

**Functions**

- int [main](#) ()

**13.258.1 Macro Definition Documentation****13.258.1.1 \_\_FFLASFFPACK\_CONFIGURATION**

```
#define __FFLASFFPACK_CONFIGURATION
```

**13.258.1.2 \_\_FFLASFFPACK\_HAVE\_LAPACK**

```
#define __FFLASFFPACK_HAVE_LAPACK 1
```

**13.258.1.3 \_\_FFLASFFPACK\_HAVE\_CLAPACK**

```
#define __FFLASFFPACK_HAVE_CLAPACK 1
```

**13.258.2 Function Documentation****13.258.2.1 main()**

```
int main (
 void)
```

**13.259 cuda.C File Reference**

```
#include <stdio.h>
#include <cuda_runtime.h>
#include <cusparse.h>
```

**Functions**

- int [main](#) ()

**13.259.1 Function Documentation****13.259.1.1 main()**

```
int main (
 void)
```

**13.260 fblas.C File Reference**

```
#include "fflas-ffpack/config-blas.h"
```

**Macros**

- #define [\\_\\_FFLASFFPACK\\_CONFIGURATION](#)

**Functions**

- void [dgemm\\_](#) (const char \*, const char \*, const int \*, const int \*, const int \*, const double \*, const double \*, const int \*, const double \*, const int \*, const double \*, double \*, const int \*)
- int [main](#) ()

### 13.260.1 Macro Definition Documentation

#### 13.260.1.1 \_\_FFLASFFPACK\_CONFIGURATION

```
#define __FFLASFFPACK_CONFIGURATION
```

### 13.260.2 Function Documentation

#### 13.260.2.1 dgemm\_()

```
void dgemm_ (
 const char * ,
 const char * ,
 const int * ,
 const int * ,
 const int * ,
 const double * ,
 const double * ,
 const int * ,
 const double * ,
 const int * ,
 const double * ,
 double * ,
 const int *)
```

#### 13.260.2.2 main()

```
int main (
 void)
```

## 13.261 lapack.C File Reference

```
#include "fflas-ffpack/config-blas.h"
```

### Macros

- [#define \\_\\_FFLASFFPACK\\_CONFIGURATION](#)
- [#define \\_\\_FFLASFFPACK\\_HAVE\\_LAPACK 1](#)

### Functions

- [int main\(\)](#)

### 13.261.1 Macro Definition Documentation

#### 13.261.1.1 \_\_FFLASFFPACK\_CONFIGURATION

```
#define __FFLASFFPACK_CONFIGURATION
```

#### 13.261.1.2 \_\_FFLASFFPACK\_HAVE\_LAPACK

```
#define __FFLASFFPACK_HAVE_LAPACK 1
```

### 13.261.2 Function Documentation

#### 13.261.2.1 main()

```
int main (
 void)
```



## 13.262 regression-check.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <givaro/modular.h>
#include "fflas-ffpack/fflas-ffpack.h"
```

### Functions

- bool [check1](#) ()
- bool [check2](#) ()
- bool [check3](#) ()
- bool [check4](#) ()
- bool [checkZeroDimCharpoly](#) ()
- bool [checkZeroDimMinPoly](#) ()
- bool [gf2ModularBalanced](#) ()
- int [main](#) ()

### 13.262.1 Function Documentation

#### 13.262.1.1 [check1\(\)](#)

```
bool check1 ()
```

#### 13.262.1.2 [check2\(\)](#)

```
bool check2 ()
```

#### 13.262.1.3 [check3\(\)](#)

```
bool check3 ()
```

#### 13.262.1.4 [check4\(\)](#)

```
bool check4 ()
```

#### 13.262.1.5 [checkZeroDimCharpoly\(\)](#)

```
bool checkZeroDimCharpoly ()
```

#### 13.262.1.6 [checkZeroDimMinPoly\(\)](#)

```
bool checkZeroDimMinPoly ()
```

#### 13.262.1.7 [gf2ModularBalanced\(\)](#)

```
bool gf2ModularBalanced ()
```

#### 13.262.1.8 [main\(\)](#)

```
int main (
 void)
```

## 13.263 test-charpoly-check.C File Reference

```
#include <iostream>
#include <stdlib.h>
#include <time.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/utils/fflas_io.h"
```

### Macros

- `#define` [ENABLE\\_CHECKER\\_charpoly](#) 1
- `#define` [TIME\\_CHECKER\\_CHARPOLY](#) 1

### Functions

- `template<class Field, class Polynomial >`  
`void printPolynomial (const Field &F, Polynomial &v)`
- `int main (int argc, char **argv)`

### 13.263.1 Macro Definition Documentation

#### 13.263.1.1 [ENABLE\\_CHECKER\\_charpoly](#)

```
#define ENABLE_CHECKER_charpoly 1
```

#### 13.263.1.2 [TIME\\_CHECKER\\_CHARPOLY](#)

```
#define TIME_CHECKER_CHARPOLY 1
```

### 13.263.2 Function Documentation

#### 13.263.2.1 [printPolynomial\(\)](#)

```
template<class Field, class Polynomial >
void printPolynomial (
 const Field & F,
 Polynomial & v)
```

#### 13.263.2.2 [main\(\)](#)

```
int main (
 int argc,
 char ** argv)
```

## 13.264 test-charpoly.C File Reference

```
#include <iostream>
#include <iomanip>
#include "givaro/modular.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/fflas_randommatrix.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include <random>
```

```
#include <chrono>
```

## Functions

- `template<class Field, class RandIter >`  
`bool launch\_test (const Field &F, size_t n, typename Field::Element *A, size_t lda, size_t nbit, RandIter &G, FFPACK::FFPACK_CHARPOLY_TAG CT)`
- `template<class Field >`  
`bool run\_with\_field (const Givaro::Integer p, uint64_t bits, size_t n, std::string file, int variant, size_t iter, uint64_t seed)`
- `int main (int argc, char **argv)`

## 13.264.1 Function Documentation

### 13.264.1.1 [launch\\_test\(\)](#)

```
template<class Field , class RandIter >
bool launch_test (
 const Field & F,
 size_t n,
 typename Field::Element * A,
 size_t lda,
 size_t nbit,
 RandIter & G,
 FFPACK::FFPACK_CHARPOLY_TAG CT)
```

### 13.264.1.2 [run\\_with\\_field\(\)](#)

```
template<class Field >
bool run_with_field (
 const Givaro::Integer p,
 uint64_t bits,
 size_t n,
 std::string file,
 int variant,
 size_t iter,
 uint64_t seed)
```

### 13.264.1.3 [main\(\)](#)

```
int main (
 int argc,
 char ** argv)
```

## 13.265 test-compressQ.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <list>
#include <vector>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include "fflas-ffpack/utils/args-parser.h"
```

## Typedefs

- typedef Givaro::Modular< double > [Field](#)

## Functions

- template<class T >  
std::ostream & [printvect](#) (std::ostream &o, vector< T > &vect)
- int [main](#) (int argc, char \*\*argv)

### 13.265.1 Typedef Documentation

#### 13.265.1.1 Field

```
typedef Givaro::Modular<double> Field
```

### 13.265.2 Function Documentation

#### 13.265.2.1 printvect()

```
template<class T >
std::ostream & printvect (
 std::ostream & o,
 vector< T > & vect)
```

[Bug](#) does not belong here

#### 13.265.2.2 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.266 test-det-check.C File Reference

```
#include <iostream>
#include <stdlib.h>
#include <time.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/checkers/checkers_ffpack.h"
#include "fflas-ffpack/checkers/checkers_ffpack.inl"
```

## Macros

- #define [ENABLE\\_CHECKER\\_Det](#) 1
- #define [TIME\\_CHECKER\\_Det](#) 1

## Functions

- int [main](#) (int argc, char \*\*argv)

### 13.266.1 Macro Definition Documentation

#### 13.266.1.1 ENABLE\_CHECKER\_Det

```
#define ENABLE_CHECKER_Det 1
```

### 13.266.1.2 TIME\_CHECKER\_Det

```
#define TIME_CHECKER_Det 1
```

## 13.266.2 Function Documentation

### 13.266.2.1 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.267 test-det.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iomanip>
#include <iostream>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
```

### Functions

- template<class [Field](#) , class RandIter >  
bool [test\\_det](#) ([Field](#) &F, size\_t n, int iter, RandIter &G)
- int [main](#) (int argc, char \*\*argv)

### 13.267.1 Function Documentation

#### 13.267.1.1 test\_det()

```
template<class Field , class RandIter >
bool test_det (
 Field & F,
 size_t n,
 int iter,
 RandIter & G)
```

[Todo](#) test with stride

#### 13.267.1.2 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.268 test-echelon.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <iomanip>
#include <givaro/modular-balanced.h>
#include <givaro/udl.h>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/ffpack/ffpack.h"
```

```
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include <random>
#include <chrono>
```

## Macros

- `#define __FFLASFFPACK_SEQUENTIAL`
- `#define __FFLASFFPACK_GAUSSJORDAN_BASECASE 25`
- `#define __FFLASFFPACK_PLUQ_THRESHOLD 25`

## Functions

- `template<class Field , class RandIter >`  
`bool test_colechelon (Field &F, size_t m, size_t n, size_t r, size_t iters, FFPACK::FFPACK_LU_TAG LuTag,`  
`RandIter &G, bool par)`
- `template<class Field , class RandIter >`  
`bool test_rowechelon (Field &F, size_t m, size_t n, size_t r, size_t iters, FFPACK::FFPACK_LU_TAG LuTag,`  
`RandIter &G, bool par)`
- `template<class Field , class RandIter >`  
`bool test_redcochelon (Field &F, size_t m, size_t n, size_t r, size_t iters, FFPACK::FFPACK_LU_TAG LuTag,`  
`RandIter &G, bool par)`
- `template<class Field , class RandIter >`  
`bool test_redrowechelon (Field &F, size_t m, size_t n, size_t r, size_t iters, FFPACK::FFPACK_LU_TAG LuTag,`  
`RandIter &G, bool par)`
- `template<class Field >`  
`bool run_with_field (Givaro::Integer q, uint64_t b, size_t m, size_t n, size_t r, size_t iters, uint64_t seed)`
- `int main (int argc, char **argv)`

## 13.268.1 Macro Definition Documentation

### 13.268.1.1 \_\_FFLASFFPACK\_SEQUENTIAL

```
#define __FFLASFFPACK_SEQUENTIAL
```

### 13.268.1.2 \_\_FFLASFFPACK\_GAUSSJORDAN\_BASECASE

```
#define __FFLASFFPACK_GAUSSJORDAN_BASECASE 25
```

### 13.268.1.3 \_\_FFLASFFPACK\_PLUQ\_THRESHOLD

```
#define __FFLASFFPACK_PLUQ_THRESHOLD 25
```

## 13.268.2 Function Documentation

### 13.268.2.1 test\_colechelon()

```
template<class Field , class RandIter >
bool test_colechelon (
 Field & F,
 size_t m,
 size_t n,
 size_t r,
 size_t iters,
 FFPACK::FFPACK_LU_TAG LuTag,
 RandIter & G,
 bool par)
```

**Todo** check Ida

### 13.268.2.2 test\_rowechelon()

```
template<class Field , class RandIter >
bool test_rowechelon (
 Field & F,
 size_t m,
 size_t n,
 size_t r,
 size_t iters,
 FFPACK::FFPACK_LU_TAG LuTag,
 RandIter & G,
 bool par)
```

**Todo** check Ida

### 13.268.2.3 test\_redcolechelon()

```
template<class Field , class RandIter >
bool test_redcolechelon (
 Field & F,
 size_t m,
 size_t n,
 size_t r,
 size_t iters,
 FFPACK::FFPACK_LU_TAG LuTag,
 RandIter & G,
 bool par)
```

**Todo** check Ida

### 13.268.2.4 test\_redrowechelon()

```
template<class Field , class RandIter >
bool test_redrowechelon (
 Field & F,
 size_t m,
 size_t n,
 size_t r,
 size_t iters,
 FFPACK::FFPACK_LU_TAG LuTag,
 RandIter & G,
 bool par)
```

**Todo** check Ida

### 13.268.2.5 run\_with\_field()

```
template<class Field >
bool run_with_field (
 Givaro::Integer q,
 uint64_t b,
 size_t m,
 size_t n,
 size_t r,
 size_t iters,
 uint64_t seed)
```

### 13.268.2.6 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.269 test-fadd.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <typeinfo>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "assert.h"
```

### Functions

- template<class [Field](#) >  
bool [test\\_fadd](#) (const [Field](#) &F, size\_t m, size\_t k, size\_t n, bool timing, uint64\_t seed)
- template<class [Field](#) >  
bool [test\\_faddin](#) (const [Field](#) &F, size\_t m, size\_t k, size\_t n, bool timing, uint64\_t seed)
- template<class [Field](#) >  
bool [test\\_fsub](#) (const [Field](#) &F, size\_t m, size\_t k, size\_t n, bool timing, uint64\_t seed)
- template<class [Field](#) >  
bool [test\\_fsubin](#) (const [Field](#) &F, size\_t m, size\_t k, size\_t n, bool timing, uint64\_t seed)
- int [main](#) (int ac, char \*\*av)

### 13.269.1 Function Documentation

#### 13.269.1.1 test\_fadd()

```
template<class Field >
bool test_fadd (
 const Field & F,
 size_t m,
 size_t k,
 size_t n,
 bool timing,
 uint64_t seed)
```

#### 13.269.1.2 test\_faddin()

```
template<class Field >
bool test_faddin (
 const Field & F,
 size_t m,
 size_t k,
 size_t n,
 bool timing,
 uint64_t seed)
```

#### 13.269.1.3 test\_fsub()

```
template<class Field >
bool test_fsub (
 const Field & F,
```



```

 size_t m,
 size_t k,
 size_t n,
 bool timing,
 uint64_t seed)

```

#### 13.269.1.4 test\_fsubin()

```

template<class Field >
bool test_fsubin (
 const Field & F,
 size_t m,
 size_t k,
 size_t n,
 bool timing,
 uint64_t seed)

```

#### 13.269.1.5 main()

```

int main (
 int ac,
 char ** av)

```

## 13.270 test-fdot.C File Reference

```

#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iomanip>
#include <iostream>
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/paladin/parallel.h"
#include "fflas-ffpack/paladin/fflas_plevel1.h"
#include <givaro/zring.h>
#include <givaro/modular.h>
#include <random>
#include <chrono>

```

### Macros

- `#define` [ENABLE\\_ALL\\_CHECKINGS](#) 1

### Functions

- `template<typename Field >`  
`bool` [check\\_fdot](#) (const [Field](#) &F, size\_t n, typename [Field::ConstElement\\_ptr](#) a, size\_t inca, typename [Field::ConstElement\\_ptr](#) b, size\_t incb)
- `template<class Field >`  
`bool` [run\\_with\\_field](#) (Givaro::Integer q, size\_t BS, size\_t n, size\_t iters, uint64\_t seed)
- `bool` [run\\_with\\_Integer](#) (size\_t BS, size\_t n, size\_t iters, uint64\_t seed)
- `int` [main](#) (int argc, char \*\*argv)

## 13.270.1 Macro Definition Documentation

### 13.270.1.1 ENABLE\_ALL\_CHECKINGS

```

#define ENABLE_ALL_CHECKINGS 1

```

## 13.270.2 Function Documentation

### 13.270.2.1 check\_fdot()

```
template<typename Field >
bool check_fdot (
 const Field & F,
 size_t n,
 typename Field::ConstElement_ptr a,
 size_t inca,
 typename Field::ConstElement_ptr b,
 size_t incb)
```

### 13.270.2.2 run\_with\_field()

```
template<class Field >
bool run_with_field (
 Givaro::Integer q,
 size_t BS,
 size_t n,
 size_t iters,
 uint64_t seed)
```

### 13.270.2.3 run\_with\_Integer()

```
bool run_with_Integer (
 size_t BS,
 size_t n,
 size_t iters,
 uint64_t seed)
```

### 13.270.2.4 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.271 test-fgemm-check.C File Reference

```
#include <iostream>
#include <stdlib.h>
#include <time.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
```

### Macros

- #define `ENABLE_ALL_CHECKINGS` 1

### Functions

- template<class `Field` , class `RandIter` >  
bool `launch_MM_dispatch` (const `Field` &F, const int mm, const int nn, const int kk, const typename `Field::Element` alpha, const typename `Field::Element` beta, const size\_t iters, `RandIter` &G)
- template<class `Field` >  
bool `run_with_field` (`Givaro::Integer` q, `uint64_t` b, int m, int n, int k, size\_t iters, `uint64_t` seed)
- int `main` (int argc, char \*\*argv)

## 13.271.1 Macro Definition Documentation

### 13.271.1.1 ENABLE\_ALL\_CHECKINGS

```
#define ENABLE_ALL_CHECKINGS 1
```

## 13.271.2 Function Documentation

### 13.271.2.1 launch\_MM\_dispatch()

```
template<class Field , class RandIter >
bool launch_MM_dispatch (
 const Field & F,
 const int mm,
 const int nn,
 const int kk,
 const typename Field::Element alpha,
 const typename Field::Element beta,
 const size_t iters,
 RandIter & G)
```

**Bug** test for ldX equal

**Bug**

**Bug** test for transpo

**Bug**

**Todo** does nbw actually do nbw recursive calls and then call blas (check ?) ?

### 13.271.2.2 run\_with\_field()

```
template<class Field >
bool run_with_field (
 Givaro::Integer q,
 uint64_t b,
 int m,
 int n,
 int k,
 size_t iters,
 uint64_t seed)
```

### 13.271.2.3 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.272 test-fgemm.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include <iomanip>
#include <iostream>
#include <givaro/modular.h>
#include <recint/rint.h>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
```

```
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include <random>
```

## Macros

- #define `ENABLE_CHECKER_fgemm` 1

## Functions

- template<class `Field` >  
bool `check_MM` (const `Field` &F, const typename `Field::Element_ptr` Cd, enum `FFLAS_TRANSPOSE` &ta, enum `FFLAS_TRANSPOSE` &tb, const size\_t m, const size\_t n, const size\_t k, const typename `Field::Element` &alpha, const typename `Field::Element_ptr` A, size\_t lda, const typename `Field::Element_ptr` B, size\_t ldb, const typename `Field::Element` &beta, const typename `Field::Element_ptr` C, size\_t ldc)
- template<class `Field` , class `RandIter` >  
bool `launch_MM` (const `Field` &F, const size\_t m, const size\_t n, const size\_t k, const typename `Field::Element` alpha, const typename `Field::Element` beta, const size\_t ldc, const size\_t lda, enum `FFLAS_TRANSPOSE` ta, const size\_t ldb, enum `FFLAS_TRANSPOSE` tb, size\_t iters, int nbw, bool par, `RandIter` &G)
- template<class `Field` , class `RandIter` >  
bool `launch_MM_dispatch` (const `Field` &F, const int mm, const int nn, const int kk, const typename `Field::Element` alpha, const typename `Field::Element` beta, const size\_t iters, const int nbw, const bool par, `RandIter` &G)
- template<class `Field` >  
bool `run_with_field` (`Givaro::Integer` q, uint64\_t b, int m, int n, int k, int nbw, size\_t iters, bool par, size\_t seed)
- int `main` (int argc, char \*\*argv)

## 13.272.1 Macro Definition Documentation

### 13.272.1.1 `ENABLE_CHECKER_fgemm`

```
#define ENABLE_CHECKER_fgemm 1
```

## 13.272.2 Function Documentation

### 13.272.2.1 `check_MM()`

```
template<class Field >
bool check_MM (
 const Field & F,
 const typename Field::Element_ptr Cd,
 enum FFLAS_TRANSPOSE & ta,
 enum FFLAS_TRANSPOSE & tb,
 const size_t m,
 const size_t n,
 const size_t k,
 const typename Field::Element & alpha,
 const typename Field::Element_ptr A,
 size_t lda,
 const typename Field::Element_ptr B,
 size_t ldb,
 const typename Field::Element & beta,
 const typename Field::Element_ptr C,
 size_t ldc)
```

### 13.272.2.2 `launch_MM()`

```
template<class Field , class RandIter >
bool launch_MM (
```

```

 const Field & F,
 const size_t m,
 const size_t n,
 const size_t k,
 const typename Field::Element alpha,
 const typename Field::Element beta,
 const size_t ldc,
 const size_t lda,
 enum FFLAS_TRANSPOSE ta,
 const size_t ldb,
 enum FFLAS_TRANSPOSE tb,
 size_t iters,
 int nbw,
 bool par,
 RandIter & G)

```

### 13.272.2.3 launch\_MM\_dispatch()

```

template<class Field , class RandIter >
bool launch_MM_dispatch (
 const Field & F,
 const int mm,
 const int nn,
 const int kk,
 const typename Field::Element alpha,
 const typename Field::Element beta,
 const size_t iters,
 const int nbw,
 const bool par,
 RandIter & G)

```

**Bug** test for ldX equal

**Bug**

**Bug** test for transpo

**Bug**

**Todo** does nbw actually do nbw recursive calls and then call blas (check ?) ?

### 13.272.2.4 run\_with\_field()

```

template<class Field >
bool run_with_field (
 Givaro::Integer q,
 uint64_t b,
 int m,
 int n,
 int k,
 int nbw,
 size_t iters,
 bool par,
 size_t seed)

```

### 13.272.2.5 main()

```

int main (
 int argc,
 char ** argv)

```

## 13.273 test-fgemv.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iomanip>
#include <iostream>
#include <givaro/modular.h>
#include <recint/rint.h>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
```

### Functions

- template<class [Field](#) >  
bool [check\\_MV](#) (const [Field](#) &F, const typename [Field::Element\\_ptr](#) Cd, enum [FFLAS\\_TRANSPOSE](#) &ta, const size\_t m, const size\_t k, const typename [Field::Element](#) &alpha, const typename [Field::Element\\_ptr](#) A, size\_t lda, const typename [Field::Element\\_ptr](#) X, size\_t incX, const typename [Field::Element](#) &beta, const typename [Field::Element\\_ptr](#) Y, size\_t incY)
- template<class [Field](#) , class RandIter >  
bool [launch\\_MV](#) (const [Field](#) &F, const size\_t m, const size\_t k, const typename [Field::Element](#) alpha, const typename [Field::Element](#) beta, const size\_t lda, enum [FFLAS\\_TRANSPOSE](#) ta, const size\_t incX, const size\_t incY, size\_t iters, bool par, RandIter &G)
- template<class [Field](#) , class RandIter >  
bool [launch\\_MV\\_dispatch](#) (const [Field](#) &F, const int mm, const int kk, const typename [Field::Element](#) alpha, const typename [Field::Element](#) beta, const size\_t iters, const bool par, RandIter &G)
- template<class [Field](#) >  
bool [run\\_with\\_field](#) (Givaro::Integer q, uint64\_t b, int m, int k, size\_t iters, bool par, uint64\_t seed)
- int [main](#) (int argc, char \*\*argv)

### 13.273.1 Function Documentation

#### 13.273.1.1 [check\\_MV\(\)](#)

```
template<class Field >
bool check_MV (
 const Field & F,
 const typename Field::Element_ptr Cd,
 enum FFLAS_TRANSPOSE & ta,
 const size_t m,
 const size_t k,
 const typename Field::Element & alpha,
 const typename Field::Element_ptr A,
 size_t lda,
 const typename Field::Element_ptr X,
 size_t incX,
 const typename Field::Element & beta,
 const typename Field::Element_ptr Y,
 size_t incY)
```

#### 13.273.1.2 [launch\\_MV\(\)](#)

```
template<class Field , class RandIter >
bool launch_MV (
 const Field & F,
 const size_t m,
 const size_t k,
 const typename Field::Element alpha,
```

```

 const typename Field::Element beta,
 const size_t lda,
 enum FFLAS_TRANSPOSE ta,
 const size_t incX,
 const size_t incY,
 size_t iters,
 bool par,
 RandIter & G)

```

#### 13.273.1.3 launch\_MV\_dispatch()

```

template<class Field , class RandIter >
bool launch_MV_dispatch (
 const Field & F,
 const int mm,
 const int kk,
 const typename Field::Element alpha,
 const typename Field::Element beta,
 const size_t iters,
 const bool par,
 RandIter & G)

```

#### 13.273.1.4 run\_with\_field()

```

template<class Field >
bool run_with_field (
 Givaro::Integer q,
 uint64_t b,
 int m,
 int k,
 size_t iters,
 bool par,
 uint64_t seed)

```

#### 13.273.1.5 main()

```

int main (
 int argc,
 char ** argv)

```

## 13.274 test-fger.C File Reference

```

#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iomanip>
#include <iostream>
#include <givaro/modular-integral.h>
#include <givaro/modular-balanced.h>
#include <givaro/givintprime.h>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/utils/fflas_io.h"

```

### Macros

- #define TIME 1

## Functions

- `template<class Field >`  
`bool check_fger (const Field &F, const typename Field::Element_ptr Cd, const size_t m, const size_t n, const`  
`typename Field::Element &alpha, const typename Field::Element_ptr x, const size_t incx, const typename`  
`Field::Element_ptr y, const size_t incy, const typename Field::Element_ptr C, const size_t ldc)`
- `template<class Field , class RandIter >`  
`bool launch_fger (const Field &F, const size_t m, const size_t n, const typename Field::Element alpha, const`  
`size_t ldc, const size_t inca, const size_t incb, size_t iters, RandIter &G)`
- `template<class Field , class RandIter >`  
`bool launch_fger_dispatch (const Field &F, const size_t nn, const typename Field::Element alpha, const`  
`size_t iters, RandIter &G)`
- `template<class Field >`  
`bool run_with_field (int64_t q, uint64_t b, size_t n, size_t iters, uint64_t seed)`
- `int main (int argc, char **argv)`

## 13.274.1 Macro Definition Documentation

### 13.274.1.1 TIME

```
#define TIME 1
```

## 13.274.2 Function Documentation

### 13.274.2.1 check\_fger()

```
template<class Field >
bool check_fger (
 const Field & F,
 const typename Field::Element_ptr Cd,
 const size_t m,
 const size_t n,
 const typename Field::Element & alpha,
 const typename Field::Element_ptr x,
 const size_t incx,
 const typename Field::Element_ptr y,
 const size_t incy,
 const typename Field::Element_ptr C,
 const size_t ldc)
```

### 13.274.2.2 launch\_fger()

```
template<class Field , class RandIter >
bool launch_fger (
 const Field & F,
 const size_t m,
 const size_t n,
 const typename Field::Element alpha,
 const size_t ldc,
 const size_t inca,
 const size_t incb,
 size_t iters,
 RandIter & G)
```

### 13.274.2.3 launch\_fger\_dispatch()

```
template<class Field , class RandIter >
bool launch_fger_dispatch (
 const Field & F,
 const size_t nn,
```



```
const typename Field::Element alpha,
const size_t iters,
RandIter & G)
```

**Bug** test for incx equal

**Bug**

**Bug** test for transpo

**Bug**

**Todo** does nbw actually do nbw recursive calls and then call blas (check ?) ?

#### 13.274.2.4 run\_with\_field()

```
template<class Field >
bool run_with_field (
 int64_t q,
 uint64_t b,
 size_t n,
 size_t iters,
 uint64_t seed)
```

#### 13.274.2.5 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.275 test-fgesv.C File Reference

```
#include <iostream>
#include <iomanip>
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include <givaro/modular.h>
```

### Functions

- template<class Field, class RandIter >  
bool **test\_square\_fgesv** (Field &F, FFLAS\_SIDE side, string fileA, string fileB, size\_t m, size\_t k, size\_t r, RandIter &G)
- template<class Field, class RandIter >  
bool **test\_rect\_fgesv** (Field &F, FFLAS\_SIDE side, string fileA, string fileB, size\_t m, size\_t n, size\_t k, size\_t r, RandIter &G)
- template<class Field >  
bool **run\_with\_field** (Givaro::Integer q, uint64\_t b, size\_t m, size\_t n, size\_t k, size\_t r, size\_t iters, string fileA, string fileB, uint64\_t &seed)
- int **main** (int argc, char \*\*argv)

## 13.275.1 Function Documentation

### 13.275.1.1 test\_square\_fgesv()

```
template<class Field , class RandIter >
bool test_square_fgesv (
 Field & F,
 FFLAS_SIDE side,
 string fileA,
 string fileB,
 size_t m,
 size_t k,
 size_t r,
 RandIter & G)
```

### 13.275.1.2 test\_rect\_fgesv()

```
template<class Field , class RandIter >
bool test_rect_fgesv (
 Field & F,
 FFLAS_SIDE side,
 string fileA,
 string fileB,
 size_t m,
 size_t n,
 size_t k,
 size_t r,
 RandIter & G)
```

### 13.275.1.3 run\_with\_field()

```
template<class Field >
bool run_with_field (
 Givaro::Integer q,
 uint64_t b,
 size_t m,
 size_t n,
 size_t k,
 size_t r,
 size_t iters,
 string fileA,
 string fileB,
 uint64_t & seed)
```

### 13.275.1.4 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.276 test-finit.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <typeinfo>
#include <givaro/modular.h>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/args-parser.h"
```

```
#include "fflas-ffpack/utils/test-utils.h"
#include "assert.h"
#include <random>
#include <chrono>
```

## Functions

- template<class [Field](#) >  
bool [test\\_freduce](#) (const [Field](#) &F, size\_t m, size\_t k, size\_t n, bool timing, uint64\_t seed)
- template<class [Field](#) >  
bool [run\\_with\\_field](#) (Givaro::Integer q, size\_t b, size\_t m, size\_t k, size\_t n, size\_t iters, bool timing, uint64\_t seed)
- int [main](#) (int ac, char \*\*av)

## 13.276.1 Function Documentation

### 13.276.1.1 test\_freduce()

```
template<class Field >
bool test_freduce (
 const Field & F,
 size_t m,
 size_t k,
 size_t n,
 bool timing,
 uint64_t seed)
```

### 13.276.1.2 run\_with\_field()

```
template<class Field >
bool run_with_field (
 Givaro::Integer q,
 size_t b,
 size_t m,
 size_t k,
 size_t n,
 size_t iters,
 bool timing,
 uint64_t seed)
```

### 13.276.1.3 main()

```
int main (
 int ac,
 char ** av)
```

## 13.277 test-fscal.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <typeinfo>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "assert.h"
```

## Functions

- `template<class Field , class RandIter >`  
`bool test_fscal (const Field &F, const typename Field::Element &alpha, size_t m, size_t k, size_t n, bool timing, RandIter &G)`
- `template<class Field >`  
`bool test_fscal (const Field &F, size_t m, size_t k, size_t n, bool timing, uint64_t seed)`
- `template<class Field , class RandIter >`  
`bool test_fscalin (const Field &F, const typename Field::Element &alpha, size_t m, size_t k, size_t n, bool timing, RandIter &G)`
- `template<class Field >`  
`bool test_fscalin (const Field &F, size_t m, size_t k, size_t n, bool timing, uint64_t seed)`
- `int main (int ac, char **av)`

## 13.277.1 Function Documentation

### 13.277.1.1 test\_fscal() [1/2]

```
template<class Field , class RandIter >
bool test_fscal (
 const Field & F,
 const typename Field::Element & alpha,
 size_t m,
 size_t k,
 size_t n,
 bool timing,
 RandIter & G)
```

### 13.277.1.2 test\_fscal() [2/2]

```
template<class Field >
bool test_fscal (
 const Field & F,
 size_t m,
 size_t k,
 size_t n,
 bool timing,
 uint64_t seed)
```

### 13.277.1.3 test\_fscalin() [1/2]

```
template<class Field , class RandIter >
bool test_fscalin (
 const Field & F,
 const typename Field::Element & alpha,
 size_t m,
 size_t k,
 size_t n,
 bool timing,
 RandIter & G)
```

### 13.277.1.4 test\_fscalin() [2/2]

```
template<class Field >
bool test_fscalin (
 const Field & F,
 size_t m,
 size_t k,
 size_t n,
```

```
bool timing,
uint64_t seed)
```

### 13.277.1.5 main()

```
int main (
 int ac,
 char ** av)
```

## 13.278 test-fsyr2k.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iomanip>
#include <iostream>
#include <random>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include <givaro/modular.h>
```

### Macros

- #define [ENABLE\\_ALL\\_CHECKINGS](#) 1

### Functions

- template<typename [Field](#) , class RandIter >  
bool [check\\_fsyr2k](#) (const [Field](#) &F, size\_t n, size\_t k, const typename [Field::Element](#) &alpha, const typename [Field::Element](#) &beta, [FFLAS::FFLAS\\_UPLO](#) uplo, [FFLAS::FFLAS\\_TRANSPOSE](#) trans, RandIter &Rand)
- template<class [Field](#) >  
bool [run\\_with\\_field](#) (Givaro::Integer q, size\_t b, size\_t n, size\_t k, int a, int c, size\_t iters, uint64\_t seed)
- int [main](#) (int argc, char \*\*argv)

## 13.278.1 Macro Definition Documentation

### 13.278.1.1 ENABLE\_ALL\_CHECKINGS

```
#define ENABLE_ALL_CHECKINGS 1
```

## 13.278.2 Function Documentation

### 13.278.2.1 check\_fsyr2k()

```
template<typename Field , class RandIter >
bool check_fsyr2k (
 const Field & F,
 size_t n,
 size_t k,
 const typename Field::Element & alpha,
 const typename Field::Element & beta,
 FFLAS::FFLAS_UPLO uplo,
 FFLAS::FFLAS_TRANSPOSE trans,
 RandIter & Rand)
```

### 13.278.2.2 run\_with\_field()

```
template<class Field >
bool run_with_field (
 Givaro::Integer q,
 size_t b,
 size_t n,
 size_t k,
 int a,
 int c,
 size_t iters,
 uint64_t seed)
```

### 13.278.2.3 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.279 test-fsyrrk.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iomanip>
#include <iostream>
#include <random>
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include <givaro/modular.h>
```

### Macros

- #define `ENABLE_ALL_CHECKINGS` 1

### Functions

- template<typename `Field` , class `RandIter` >  
bool `check_fsyrrk` (const `Field` &F, size\_t n, size\_t k, size\_t w, const typename `Field::Element` &alpha, const typename `Field::Element` &beta, `FFLAS::FFLAS_UPLO` uplo, `FFLAS::FFLAS_TRANSPOSE` trans, `RandIter` &Rand)
- template<typename `Field` , class `RandIter` >  
bool `check_fsyrrk_diag` (const `Field` &F, size\_t n, size\_t k, const typename `Field::Element` &alpha, const typename `Field::Element` &beta, `FFLAS::FFLAS_UPLO` uplo, `FFLAS::FFLAS_TRANSPOSE` trans, `RandIter` &Rand)
- template<typename `Field` , class `RandIter` >  
bool `check_fsyrrk_bkdiag` (const `Field` &F, size\_t n, size\_t k, const typename `Field::Element` &alpha, const typename `Field::Element` &beta, `FFLAS_UPLO` uplo, `FFLAS_TRANSPOSE` trans, `RandIter` &Rand)
- template<class `Field` , class `RandIter` >  
bool `check_computeS1S2` (const `Field` &F, size\_t N, size\_t K, `FFLAS_TRANSPOSE` trans, `RandIter` &G)
- template<class `Field` >  
bool `run_with_field` (Givaro::Integer q, size\_t b, size\_t n, size\_t k, size\_t w, int a, int c, size\_t iters, uint64\_t seed)
- int `main` (int argc, char \*\*argv)

## 13.279.1 Macro Definition Documentation

### 13.279.1.1 ENABLE\_ALL\_CHECKINGS

```
#define ENABLE_ALL_CHECKINGS 1
```

## 13.279.2 Function Documentation

### 13.279.2.1 check\_fsyrc()

```
template<typename Field , class RandIter >
bool check_fsyrc (
 const Field & F,
 size_t n,
 size_t k,
 size_t w,
 const typename Field::Element & alpha,
 const typename Field::Element & beta,
 FFLAS::FFLAS_UPLO uplo,
 FFLAS::FFLAS_TRANSPOSE trans,
 RandIter & Rand)
```

### 13.279.2.2 check\_fsyrc\_diag()

```
template<typename Field , class RandIter >
bool check_fsyrc_diag (
 const Field & F,
 size_t n,
 size_t k,
 const typename Field::Element & alpha,
 const typename Field::Element & beta,
 FFLAS::FFLAS_UPLO uplo,
 FFLAS::FFLAS_TRANSPOSE trans,
 RandIter & Rand)
```

### 13.279.2.3 check\_fsyrc\_bkdiag()

```
template<typename Field , class RandIter >
bool check_fsyrc_bkdiag (
 const Field & F,
 size_t n,
 size_t k,
 const typename Field::Element & alpha,
 const typename Field::Element & beta,
 FFLAS_UPLO uplo,
 FFLAS_TRANSPOSE trans,
 RandIter & Rand)
```

### 13.279.2.4 check\_computeS1S2()

```
template<class Field , class RandIter >
bool check_computeS1S2 (
 const Field & F,
 size_t N,
 size_t K,
 FFLAS_TRANSPOSE trans,
 RandIter & G)
```

### 13.279.2.5 run\_with\_field()

```
template<class Field >
bool run_with_field (
 Givaro::Integer q,
 size_t b,
 size_t n,
 size_t k,
 size_t w,
 int a,
 int c,
 size_t iters,
 uint64_t seed)
```

### 13.279.2.6 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.280 test-fsytrf.C File Reference

```
#include <iostream>
#include <iterator>
#include <vector>
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include "fflas-ffpack/utils/args-parser.h"
#include <iomanip>
#include <random>
#include <chrono>
#include <givaro/modular.h>
#include "fflas-ffpack/utils/test-utils.h"
```

### Functions

- template<typename T >  
std::ostream & [operator<<](#) (std::ostream &os, const std::vector< T > &x)
- template<class Field , class Randlter >  
bool [test\\_RPM\\_fsytrf](#) (Field &F, FFLAS\_UPLO uplo, string file, size\_t n, size\_t r, Randlter &G, size\_t threshold)
- template<class Field , class Randlter >  
bool [test\\_generic\\_fsytrf](#) (Field &F, FFLAS\_UPLO uplo, string file, size\_t n, Randlter &G, size\_t threshold)
- template<class Field >  
bool [run\\_with\\_field](#) (Givaro::Integer q, uint64\_t b, size\_t n, size\_t r, size\_t iters, string file, size\_t threshold, uint64\_t &seed)
- int [main](#) (int argc, char \*\*argv)

### 13.280.1 Function Documentation

#### 13.280.1.1 operator<<()

```
template<typename T >
std::ostream & operator<< (
 std::ostream & os,
 const std::vector< T > & x)
```



**13.280.1.2 test\_RPM\_fsytrf()**

```
template<class Field , class RandIter >
bool test_RPM_fsytrf (
 Field & F,
 FFLAS_UPLO uplo,
 string file,
 size_t n,
 size_t r,
 RandIter & G,
 size_t threshold)
```

**13.280.1.3 test\_generic\_fsytrf()**

```
template<class Field , class RandIter >
bool test_generic_fsytrf (
 Field & F,
 FFLAS_UPLO uplo,
 string file,
 size_t n,
 RandIter & G,
 size_t threshold)
```

**13.280.1.4 run\_with\_field()**

```
template<class Field >
bool run_with_field (
 Givaro::Integer q,
 uint64_t b,
 size_t n,
 size_t r,
 size_t iters,
 string file,
 size_t threshold,
 uint64_t & seed)
```

**13.280.1.5 main()**

```
int main (
 int argc,
 char ** argv)
```

**13.281 test-fftrmm.C File Reference**

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <givaro/modular-integer.h>
#include <iomanip>
#include <iostream>
#include <random>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include <givaro/modular.h>
#include <givaro/modular-balanced.h>
```

## Macros

- `#define __FFLASFFPACK_SEQUENTIAL`

## Functions

- `template<typename Field, class RandIter >`  
`bool check_ftmm (const Field &F, size_t m, size_t n, const typename Field::Element &alpha,`  
`FFLAS::FFLAS_SIDE side, FFLAS::FFLAS_UPLO uplo, FFLAS::FFLAS_TRANSPOSE trans, FFLAS::FFLAS_DIAG`  
`diag, RandIter &Rand)`
- `template<class Field >`  
`bool run_with_field (Givaro::Integer q, size_t b, size_t m, size_t n, uint64_t a, size_t iters, uint64_t seed)`
- `int main (int argc, char **argv)`

## 13.281.1 Macro Definition Documentation

### 13.281.1.1 \_\_FFLASFFPACK\_SEQUENTIAL

```
#define __FFLASFFPACK_SEQUENTIAL
```

## 13.281.2 Function Documentation

### 13.281.2.1 check\_ftmm()

```
template<typename Field, class RandIter >
bool check_ftmm (
 const Field & F,
 size_t m,
 size_t n,
 const typename Field::Element & alpha,
 FFLAS::FFLAS_SIDE side,
 FFLAS::FFLAS_UPLO uplo,
 FFLAS::FFLAS_TRANSPOSE trans,
 FFLAS::FFLAS_DIAG diag,
 RandIter & Rand)
```

### 13.281.2.2 run\_with\_field()

```
template<class Field >
bool run_with_field (
 Givaro::Integer q,
 size_t b,
 size_t m,
 size_t n,
 uint64_t a,
 size_t iters,
 uint64_t seed)
```

### 13.281.2.3 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.282 test-ftmmv.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iomanip>
#include <iostream>
```

```
#include <chrono>
#include <random>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include <givaro/modular.h>
```

## Macros

- `#define __FFLASFFPACK_SEQUENTIAL`
- `#define ENABLE_ALL_CHECKINGS 1`

## Functions

- `template<typename Field , class RandIter >`  
`bool check_ffmv (const Field &F, size_t n, FFLAS_UPLO uplo, FFLAS_TRANSPOSE trans, FFLAS_DIAG diag, RandIter &Rand)`
- `template<class Field >`  
`bool run_with_field (Givaro::Integer q, size_t b, size_t n, size_t iters, uint64_t seed)`
- `int main (int argc, char **argv)`

## 13.282.1 Macro Definition Documentation

### 13.282.1.1 \_\_FFLASFFPACK\_SEQUENTIAL

```
#define __FFLASFFPACK_SEQUENTIAL
```

### 13.282.1.2 ENABLE\_ALL\_CHECKINGS

```
#define ENABLE_ALL_CHECKINGS 1
```

## 13.282.2 Function Documentation

### 13.282.2.1 check\_ffmv()

```
template<typename Field , class RandIter >
bool check_ffmv (
 const Field & F,
 size_t n,
 FFLAS_UPLO uplo,
 FFLAS_TRANSPOSE trans,
 FFLAS_DIAG diag,
 RandIter & Rand)
```

### 13.282.2.2 run\_with\_field()

```
template<class Field >
bool run_with_field (
 Givaro::Integer q,
 size_t b,
 size_t n,
 size_t iters,
 uint64_t seed)
```

**13.282.2.3 main()**

```
int main (
 int argc,
 char ** argv)
```

**13.283 test-ftsrm-check.C File Reference**

```
#include <iostream>
#include <stdlib.h>
#include <time.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/utils/fflas_io.h"
```

**Macros**

- `#define` [ENABLE\\_ALL\\_CHECKINGS](#) 1

**Functions**

- `int` [main](#) (int argc, char \*\*argv)

**13.283.1 Macro Definition Documentation****13.283.1.1 ENABLE\_ALL\_CHECKINGS**

```
#define ENABLE_ALL_CHECKINGS 1
```

**13.283.2 Function Documentation****13.283.2.1 main()**

```
int main (
 int argc,
 char ** argv)
```

**13.284 test-ftsrm.C File Reference**

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <givaro/modular-integer.h>
#include <iomanip>
#include <iostream>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include <givaro/modular.h>
#include <givaro/modular-balanced.h>
#include <random>
```

**Macros**

- `#define` [\\_\\_FFLASFFPACK\\_SEQUENTIAL](#)
- `#define` [ENABLE\\_ALL\\_CHECKINGS](#) 1

## Functions

- `template<typename Field , class RandIter >`  
`bool check_ftrsm (const Field &F, size_t m, size_t n, const typename Field::Element &alpha,`  
`FFLAS::FFLAS_SIDE side, FFLAS::FFLAS_UPLO uplo, FFLAS::FFLAS_TRANSPOSE trans, FFLAS::FFLAS_DIAG`  
`diag, RandIter &Rand)`
- `template<class Field >`  
`bool run_with_field (Givaro::Integer q, size_t b, size_t m, size_t n, uint64_t a, size_t iters, uint64_t seed)`
- `int main (int argc, char **argv)`

## 13.284.1 Macro Definition Documentation

### 13.284.1.1 \_\_FFLASFFPACK\_SEQUENTIAL

```
#define __FFLASFFPACK_SEQUENTIAL
```

### 13.284.1.2 ENABLE\_ALL\_CHECKINGS

```
#define ENABLE_ALL_CHECKINGS 1
```

## 13.284.2 Function Documentation

### 13.284.2.1 check\_ftrsm()

```
template<typename Field , class RandIter >
bool check_ftrsm (
 const Field & F,
 size_t m,
 size_t n,
 const typename Field::Element & alpha,
 FFLAS::FFLAS_SIDE side,
 FFLAS::FFLAS_UPLO uplo,
 FFLAS::FFLAS_TRANSPOSE trans,
 FFLAS::FFLAS_DIAG diag,
 RandIter & Rand)
```

### 13.284.2.2 run\_with\_field()

```
template<class Field >
bool run_with_field (
 Givaro::Integer q,
 size_t b,
 size_t m,
 size_t n,
 uint64_t a,
 size_t iters,
 uint64_t seed)
```

### 13.284.2.3 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.285 test-ffrssyr2k.C File Reference

```
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <givaro/modular-integer.h>
```

```
#include <iomanip>
#include <iostream>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include <givaro/modular.h>
#include <givaro/modular-balanced.h>
#include <random>
```

## Macros

- #define [ENABLE\\_ALL\\_CHECKINGS](#) 1

## Functions

- template<typename [Field](#) , class RandIter >  
bool [check\\_ftrssyr2k](#) (const [Field](#) &F, size\_t n, [FFLAS::FFLAS\\_UPLO](#) uplo, [FFLAS::FFLAS\\_DIAG](#) diagA, RandIter &Rand)
- template<class [Field](#) >  
bool [run\\_with\\_field](#) (Givaro::Integer q, size\_t b, size\_t n, size\_t iters, uint64\_t seed)
- int [main](#) (int argc, char \*\*argv)

## 13.285.1 Macro Definition Documentation

### 13.285.1.1 [ENABLE\\_ALL\\_CHECKINGS](#)

```
#define ENABLE_ALL_CHECKINGS 1
```

## 13.285.2 Function Documentation

### 13.285.2.1 [check\\_ftrssyr2k\(\)](#)

```
template<typename Field , class RandIter >
bool check_ftrssyr2k (
 const Field & F,
 size_t n,
 FFLAS::FFLAS_UPLO uplo,
 FFLAS::FFLAS_DIAG diagA,
 RandIter & Rand)
```

### 13.285.2.2 [run\\_with\\_field\(\)](#)

```
template<class Field >
bool run_with_field (
 Givaro::Integer q,
 size_t b,
 size_t n,
 size_t iters,
 uint64_t seed)
```

### 13.285.2.3 [main\(\)](#)

```
int main (
 int argc,
 char ** argv)
```

## 13.286 test-ftrstr.C File Reference

```
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <givaro/modular-integer.h>
#include <iomanip>
#include <iostream>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include <givaro/modular.h>
#include <givaro/modular-balanced.h>
#include <random>
```

### Macros

- `#define` [ENABLE\\_ALL\\_CHECKINGS](#) 1

### Functions

- `template<typename Field , class RandIter >`  
`bool check\_ftrstr (const Field &F, size_t n, FFLAS::FFLAS\_SIDE side, FFLAS::FFLAS\_UPLO uplo,  
FFLAS::FFLAS\_DIAG diagA, FFLAS::FFLAS\_DIAG diagB, RandIter &Rand)`
- `template<class Field >`  
`bool run\_with\_field (Givaro::Integer q, size_t b, size_t n, size_t iters, uint64_t seed)`
- `int main (int argc, char **argv)`

## 13.286.1 Macro Definition Documentation

### 13.286.1.1 [ENABLE\\_ALL\\_CHECKINGS](#)

```
#define ENABLE_ALL_CHECKINGS 1
```

## 13.286.2 Function Documentation

### 13.286.2.1 [check\\_ftrstr\(\)](#)

```
template<typename Field , class RandIter >
bool check_ftrstr (
 const Field & F,
 size_t n,
 FFLAS::FFLAS_SIDE side,
 FFLAS::FFLAS_UPLO uplo,
 FFLAS::FFLAS_DIAG diagA,
 FFLAS::FFLAS_DIAG diagB,
 RandIter & Rand)
```

### 13.286.2.2 [run\\_with\\_field\(\)](#)

```
template<class Field >
bool run_with_field (
 Givaro::Integer q,
 size_t b,
 size_t n,
 size_t iters,
 uint64_t seed)
```

### 13.286.2.3 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.287 test-fftrsv.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iomanip>
#include <iostream>
#include <random>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include <givaro/modular.h>
```

### Macros

- `#define __FFLASFFPACK_SEQUENTIAL`
- `#define ENABLE_ALL_CHECKINGS 1`

### Functions

- `template<typename Field , class RandIter >`  
`bool check_fftrsv (const Field &F, size_t n, FFLAS_UPLO uplo, FFLAS_TRANSPOSE trans, FFLAS_DIAG diag, RandIter &Rand)`
- `template<class Field >`  
`bool run_with_field (Givaro::Integer q, size_t b, size_t n, size_t iters, uint64_t seed)`
- `int main (int argc, char **argv)`

## 13.287.1 Macro Definition Documentation

### 13.287.1.1 \_\_FFLASFFPACK\_SEQUENTIAL

```
#define __FFLASFFPACK_SEQUENTIAL
```

### 13.287.1.2 ENABLE\_ALL\_CHECKINGS

```
#define ENABLE_ALL_CHECKINGS 1
```

## 13.287.2 Function Documentation

### 13.287.2.1 check\_fftrsv()

```
template<typename Field , class RandIter >
bool check_fftrsv (
 const Field & F,
 size_t n,
 FFLAS_UPLO uplo,
 FFLAS_TRANSPOSE trans,
 FFLAS_DIAG diag,
 RandIter & Rand)
```



**13.287.2.2 run\_with\_field()**

```
template<class Field >
bool run_with_field (
 Givaro::Integer q,
 size_t b,
 size_t n,
 size_t iters,
 uint64_t seed)
```

**13.287.2.3 main()**

```
int main (
 int argc,
 char ** argv)
```

**13.288 test-fftrtri.C File Reference**

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iomanip>
#include <iostream>
#include <random>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include <givaro/modular.h>
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/ffpack/ffpack.h"
```

**Macros**

- #define [\\_\\_FFLASFFPACK\\_SEQUENTIAL](#)
- #define [ENABLE\\_ALL\\_CHECKINGS](#) 1

**Functions**

- template<typename [Field](#) , class RandIter >  
bool [check\\_fftrtri](#) (const [Field](#) &F, size\_t n, [FFLAS\\_UPLO](#) uplo, [FFLAS\\_DIAG](#) diag, RandIter &Rand)
- template<class [Field](#) >  
bool [run\\_with\\_field](#) (Givaro::Integer q, size\_t b, size\_t n, size\_t iters, uint64\_t seed)
- int [main](#) (int argc, char \*\*argv)

**13.288.1 Macro Definition Documentation****13.288.1.1 \_\_FFLASFFPACK\_SEQUENTIAL**

```
#define __FFLASFFPACK_SEQUENTIAL
```

**13.288.1.2 ENABLE\_ALL\_CHECKINGS**

```
#define ENABLE_ALL_CHECKINGS 1
```

**13.288.2 Function Documentation****13.288.2.1 check\_fftrtri()**

```
template<typename Field , class RandIter >
bool check_fftrtri (
```

```

 const Field & F,
 size_t n,
 FFLAS_UPLO uplo,
 FFLAS_DIAG diag,
 RandIter & Rand)

```

### 13.288.2.2 run\_with\_field()

```

template<class Field >
bool run_with_field (
 Givaro::Integer q,
 size_t b,
 size_t n,
 size_t iters,
 uint64_t seed)

```

### 13.288.2.3 main()

```

int main (
 int argc,
 char ** argv)

```

## 13.289 test-interfaces-c.c File Reference

```

#include <fflas-ffpack/interfaces/libs/fflas_c.h>
#include <fflas-ffpack/interfaces/libs/ffpack_c.h>
#include <stdlib.h>
#include <stdio.h>

```

### Functions

- int [main](#) ()

### 13.289.1 Function Documentation

#### 13.289.1.1 main()

```

int main (
 void)

```

## 13.290 test-invert-check.C File Reference

```

#include <iostream>
#include <stdlib.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/utils/fflas_randommatrix.h"

```

### Macros

- #define [ENABLE\\_ALL\\_CHECKINGS](#) 1

### Functions

- int [main](#) (int argc, char \*\*argv)

## 13.290.1 Macro Definition Documentation

### 13.290.1.1 ENABLE\_ALL\_CHECKINGS

```
#define ENABLE_ALL_CHECKINGS 1
```

## 13.290.2 Function Documentation

### 13.290.2.1 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.291 test-io.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <random>
#include <givaro/modular.h>
#include <givaro/zring.h>
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/args-parser.h"
```

### Data Structures

- struct [CompactElement](#)< [Element](#) >
- struct [CompactElement](#)< [double](#) >
- struct [CompactElement](#)< [float](#) >
- struct [CompactElement](#)< [int64\\_t](#) >
- struct [CompactElement](#)< [int32\\_t](#) >
- struct [CompactElement](#)< [int16\\_t](#) >

### Functions

- template<class [Field](#) >  
bool [run\\_with\\_field](#) (Givaro::Integer q, uint64\_t b, size\_t m, size\_t n, size\_t iters, uint64\_t seed)
- int [main](#) (int argc, char \*\*argv)

## 13.291.1 Function Documentation

### 13.291.1.1 run\_with\_field()

```
template<class Field >
bool run_with_field (
 Givaro::Integer q,
 uint64_t b,
 size_t m,
 size_t n,
 size_t iters,
 uint64_t seed)
```

### 13.291.1.2 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.292 test-lu.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <givaro/modular-balanced.h>
#include <iostream>
#include <iomanip>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/utils/args-parser.h"
#include <random>
```

### Macros

- #define [BASECASE\\_K](#) 37
- #define [\\_\\_FFLASFFPACK\\_SEQUENTIAL](#)
- #define [\\_\\_LUDIVINE\\_CUTOFF](#) 1

### Functions

- template<class [Field](#), [FFLAS\\_DIAG](#) diag, [FFLAS\\_TRANSPOSE](#) trans>  
bool [test\\_LUdivine](#) (const [Field](#) &F, typename [Field::ConstElement\\_ptr](#) A, size\_t lda, size\_t r, size\_t m, size\_t n)  
*Tests the LUdivine routine.*
- template<class [Field](#), [FFLAS\\_DIAG](#) diag>  
bool [verifPLUQ](#) (const [Field](#) &F, typename [Field::ConstElement\\_ptr](#) A, size\_t lda, typename [Field::Element\\_ptr](#) PLUQ, size\_t ldpluq, size\_t \*P, size\_t \*Q, size\_t m, size\_t n, size\_t R)  
*Verifies that  $B = PLUQ$  where A stores  $[L|U]$ .*
- template<class [Field](#), [FFLAS\\_DIAG](#) diag, class [RandIter](#) >  
bool [test\\_pluq](#) (const [Field](#) &F, typename [Field::ConstElement\\_ptr](#) A, size\_t r, size\_t m, size\_t n, size\_t lda, [RandIter](#) &G)  
*Tests the LUdivine routine.*
- template<class [Field](#), [FFLAS\\_DIAG](#) diag, [FFLAS\\_TRANSPOSE](#) trans, class [RandIter](#) >  
bool [launch\\_test](#) (const [Field](#) &F, size\_t r, size\_t m, size\_t n, [RandIter](#) &G)
- template<class [Field](#) >  
bool [run\\_with\\_field](#) ([Givaro::Integer](#) q, uint64\_t b, size\_t m, size\_t n, size\_t r, size\_t iters, uint64\_t seed)
- int [main](#) (int argc, char \*\*argv)

### Variables

- [Givaro::Timer](#) [tperm](#)
- [Givaro::Timer](#) [tgemm](#)
- [Givaro::Timer](#) [tBC](#)
- [Givaro::Timer](#) [ttrsm](#)
- [Givaro::Timer](#) [trest](#)
- [Givaro::Timer](#) [timtot](#)
- size\_t [mvcnt](#) = 0

## 13.292.1 Macro Definition Documentation

### 13.292.1.1 BASECASE\_K

```
#define BASECASE_K 37
```

### 13.292.1.2 \_\_FFLASFFPACK\_SEQUENTIAL

```
#define __FFLASFFPACK_SEQUENTIAL
```

### 13.292.1.3 \_\_LUDIVINE\_CUTOFF

```
#define __LUDIVINE_CUTOFF 1
```

## 13.292.2 Function Documentation

### 13.292.2.1 test\_LUdivine()

```
template<class Field , FFLAS::FFLAS_DIAG diag, FFLAS_TRANSPOSE trans>
bool test_LUdivine (
 const Field & F,
 typename Field::ConstElement_ptr A,
 size_t lda,
 size_t r,
 size_t m,
 size_t n)
```

Tests the LUdivine routine.

#### Template Parameters

|              |                    |
|--------------|--------------------|
| <i>Field</i> | Field              |
| <i>Diag</i>  | Unit diagonal in U |
| <i>Trans</i> |                    |

#### Parameters

|            |                       |
|------------|-----------------------|
| <i>F</i>   | field                 |
| <i>A</i>   | Matrix (preallocated) |
| <i>r</i>   | rank of A             |
| <i>m</i>   | rows                  |
| <i>n</i>   | cols                  |
| <i>lda</i> | leading dim of A      |

#### Returns

0 iff correct, 1 otherwise

### 13.292.2.2 verifPLUQ()

```
template<class Field , FFLAS_DIAG diag>
bool verifPLUQ (
 const Field & F,
 typename Field::ConstElement_ptr A,
 size_t lda,
 typename Field::Element_ptr PLUQ,
 size_t ldpluq,
 size_t * P,
 size_t * Q,
 size_t m,
 size_t n,
 size_t R)
```

Verifies that  $B = PLUQ$  where A stores  $[L|U]$ .

## Template Parameters

|              |                    |
|--------------|--------------------|
| <i>Field</i> | Field              |
| <i>Diag</i>  | Unit diagonal in U |

## Parameters

|            |                       |
|------------|-----------------------|
| <i>F</i>   | field                 |
| <i>A</i>   | Matrix (preallocated) |
| <i>r</i>   | rank of A             |
| <i>m</i>   | rows                  |
| <i>n</i>   | cols                  |
| <i>lda</i> | leading dim of A      |

## Returns

0 iff correct, 1 otherwise

**13.292.2.3 test\_pluq()**

```
template<class Field , FFLAS_DIAG diag, class RandIter >
bool test_pluq (
 const Field & F,
 typename Field::ConstElement_ptr A,
 size_t r,
 size_t m,
 size_t n,
 size_t lda,
 RandIter & G)
```

Tests the LUdivine routine.

## Template Parameters

|              |                    |
|--------------|--------------------|
| <i>Field</i> | Field              |
| <i>Diag</i>  | Unit diagonal in U |
| <i>Trans</i> |                    |

## Parameters

|            |                       |
|------------|-----------------------|
| <i>F</i>   | field                 |
| <i>A</i>   | Matrix (preallocated) |
| <i>r</i>   | rank of A             |
| <i>m</i>   | rows                  |
| <i>n</i>   | cols                  |
| <i>lda</i> | leading dim of A      |

## Returns

0 iff correct, 1 otherwise

**13.292.2.4 launch\_test()**

```
template<class Field , FFLAS_DIAG diag, FFLAS_TRANSPOSE trans, class RandIter >
```

```
bool launch_test (
 const Field & F,
 size_t r,
 size_t m,
 size_t n,
 RandIter & G)
```

#### 13.292.2.5 run\_with\_field()

```
template<class Field >
bool run_with_field (
 Givaro::Integer q,
 uint64_t b,
 size_t m,
 size_t n,
 size_t r,
 size_t iters,
 uint64_t seed)
```

#### 13.292.2.6 main()

```
int main (
 int argc,
 char ** argv)
```

### 13.292.3 Variable Documentation

#### 13.292.3.1 tperm

Givaro::Timer tperm

#### 13.292.3.2 tgemm

Givaro::Timer tgemm

#### 13.292.3.3 tBC

Givaro::Timer tBC

#### 13.292.3.4 ttrsm

Givaro::Timer ttrsm

#### 13.292.3.5 trest

Givaro::Timer trest

#### 13.292.3.6 timtot

Givaro::Timer timtot

#### 13.292.3.7 mvcnt

size\_t mvcnt = 0

## 13.293 test-maxdelayeddim.C File Reference

```
#include <givaro/modular.h>
#include <recint/rint.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include <stdlib.h>
#include <stdio.h>
```

### Macros

- `#define MAX_WITH_SIZE_T(x) ( (static_cast<uint64_t>(std::numeric_limits<size_t>::max()) < x)? std::numeric_limits<size_t>::max() : x )`

### Functions

- `template<class Field >`  
`bool test (Givaro::Integer p, size_t kmax)`
- `int main ()`

### 13.293.1 Macro Definition Documentation

#### 13.293.1.1 MAX\_WITH\_SIZE\_T

```
#define MAX_WITH_SIZE_T(
 x) ((static_cast<uint64_t>(std::numeric_limits<size_t>::max()) < x)? std::
::numeric_limits<size_t>::max() : x)
```

### 13.293.2 Function Documentation

#### 13.293.2.1 test()

```
template<class Field >
bool test (
 Givaro::Integer p,
 size_t kmax)
```

#### 13.293.2.2 main()

```
int main (
 void)
```

## 13.294 test-minpoly.C File Reference

```
#include <iomanip>
#include <iostream>
#include <random>
#include <chrono>
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/fflas_randommatrix.h"
#include "fflas-ffpack/utils/test-utils.h"
#include <givaro/modular.h>
#include <givaro/modular-balanced.h>
#include <givaro/modular-integer.h>
#include <givaro/givpoly1factor.h>
```



```
#include <givaro/givpoly1.h>
```

## Functions

- template<typename [Field](#) , class RandIter >  
bool [check\\_minpoly](#) (const [Field](#) &F, size\_t n, RandIter &G)
- template<class [Field](#) >  
bool [run\\_with\\_field](#) (Givaro::Integer q, size\_t b, size\_t n, size\_t iters, uint64\_t seed)
- int [main](#) (int argc, char \*\*argv)

## 13.294.1 Function Documentation

### 13.294.1.1 [check\\_minpoly\(\)](#)

```
template<typename Field , class RandIter >
bool check_minpoly (
 const Field & F,
 size_t n,
 RandIter & G)
```

### 13.294.1.2 [run\\_with\\_field\(\)](#)

```
template<class Field >
bool run_with_field (
 Givaro::Integer q,
 size_t b,
 size_t n,
 size_t iters,
 uint64_t seed)
```

### 13.294.1.3 [main\(\)](#)

```
int main (
 int argc,
 char ** argv)
```

## 13.295 test-multifile1.C File Reference

```
#include "fflas-ffpack/fflas-ffpack.h"
```

## 13.296 test-multifile2.C File Reference

```
#include "fflas-ffpack/fflas-ffpack.h"
```

## Functions

- int [main](#) (void)

## 13.296.1 Function Documentation

### 13.296.1.1 [main\(\)](#)

```
int main (
 void)
```

## 13.297 test-nullspace.C File Reference

```
#include <iomanip>
#include <iostream>
#include "fflas-ffpack/ffpack/ffpack.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/fflas_randommatrix.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/utils/timer.h"
```

### Functions

- template<class [Field](#) >  
std::string [checkingMessage](#) (const [Field](#) &F)
- template<class [Field](#) >  
[Field::Element\\_ptr](#) [readOrRandomMatrixWithRankAndRandomRPM](#) (const [Field](#) &F, std::string file, size\_t m, size\_t n, size\_t lda, size\_t r, uint64\_t seed)  
*If file is not empty, read it and set m, n, lda and r.*
- template<class [Field](#) >  
bool [test\\_nullspace](#) ([Field](#) &F, [FFLAS::FFLAS\\_SIDE](#) side, size\_t m, size\_t n, size\_t r, typename [Field::Element\\_ptr](#) A, size\_t lda)
- template<class [Field](#) >  
bool [run\\_with\\_field](#) (Givaro::Integer q, uint64\_t b, size\_t m, size\_t n, size\_t r, size\_t iters, std::string file, uint64\_t &seed)
- int [main](#) (int argc, char \*\*argv)

### 13.297.1 Function Documentation

#### 13.297.1.1 [checkingMessage\(\)](#)

```
template<class Field >
std::string checkingMessage (
 const Field & F)
```

#### 13.297.1.2 [readOrRandomMatrixWithRankAndRandomRPM\(\)](#)

```
template<class Field >
Field::Element_ptr readOrRandomMatrixWithRankAndRandomRPM (
 const Field & F,
 std::string file,
 size_t m,
 size_t n,
 size_t lda,
 size_t r,
 uint64_t seed)
```

If file is not empty, read it and set m, n, lda and r.

Otherwise, generate a random matrix of size m x n with random lda.

#### 13.297.1.3 [test\\_nullspace\(\)](#)

```
template<class Field >
bool test_nullspace (
 Field & F,
 FFLAS::FFLAS_SIDE side,
 size_t m,
 size_t n,
 size_t r,
```

```
typename Field::Element_ptr A,
size_t lda)
```

#### 13.297.1.4 run\_with\_field()

```
template<class Field >
bool run_with_field (
 Givaro::Integer q,
 uint64_t b,
 size_t m,
 size_t n,
 size_t r,
 size_t iters,
 std::string file,
 uint64_t & seed)
```

#### 13.297.1.5 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.298 test-permutations.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <iostream>
#include <givaro/modular.h>
#include "fflas-ffpack/ffpack/ffpack.h"
```

### Functions

- bool [checkMonotonicApplyP](#) (FFLAS\_SIDE Side, FFLAS\_TRANSPOSE trans, size\_t \*P, size\_t N, size\_t R)
- int [main](#) ()

### Variables

- Givaro::Timer [tperm](#)
- Givaro::Timer [tgemm](#)
- Givaro::Timer [tBC](#)
- Givaro::Timer [ttrsm](#)
- Givaro::Timer [trest](#)
- Givaro::Timer [timtot](#)

## 13.298.1 Function Documentation

### 13.298.1.1 checkMonotonicApplyP()

```
bool checkMonotonicApplyP (
 FFLAS_SIDE Side,
 FFLAS_TRANSPOSE trans,
 size_t * P,
 size_t N,
 size_t R)
```

### 13.298.1.2 main()

```
int main (
 void)
```

## 13.298.2 Variable Documentation

### 13.298.2.1 tperm

Givaro::Timer tperm

### 13.298.2.2 tgemm

Givaro::Timer tgemm

### 13.298.2.3 tBC

Givaro::Timer tBC

### 13.298.2.4 ttrsm

Givaro::Timer ttrsm

### 13.298.2.5 trest

Givaro::Timer trest

### 13.298.2.6 timtot

Givaro::Timer timtot

## 13.299 test-pluq-check.C File Reference

```
#include <iostream>
#include <stdlib.h>
#include <time.h>
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
```

### Macros

- `#define` [ENABLE\\_ALL\\_CHECKINGS](#) 1

### Functions

- `int` [main](#) (`int` argc, `char **`argv)

## 13.299.1 Macro Definition Documentation

### 13.299.1.1 ENABLE\_ALL\_CHECKINGS

```
#define ENABLE_ALL_CHECKINGS 1
```

## 13.299.2 Function Documentation

### 13.299.2.1 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.300 test-quasisep.C File Reference

```
#include "fflas-ffpack/fflas-ffpack-config.h"
#include <givaro/modular-balanced.h>
#include <iostream>
#include <iomanip>
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/utils/args-parser.h"
#include <random>
```

### Functions

- template<class [Field](#), [FFLAS\\_DIAG](#) diag, class RandIter >  
bool [test\\_BruhatGenerator](#) (const [Field](#) &F, size\_t n, size\_t r, size\_t t, typename [Field::ConstElement\\_ptr](#) A, size\_t lda, typename [Field::Element\\_ptr](#) TS, size\_t l, RandIter &G)
- template<class [Field](#), [FFLAS\\_DIAG](#) diag, class RandIter >  
bool [launch\\_test](#) (const [Field](#) &F, size\_t n, size\_t r, size\_t t, size\_t l, RandIter &G)
- template<class [Field](#), class RandGen >  
bool [testLTQSRPM](#) (const [Field](#) &F, size\_t n, size\_t r, size\_t t, RandGen &G)
- template<class [Field](#) >  
bool [run\\_with\\_field](#) (Givaro::Integer q, uint64\_t b, size\_t n, size\_t r, size\_t t, size\_t l, size\_t iters, uint64\_t seed)
- int [main](#) (int argc, char \*\*argv)

### 13.300.1 Function Documentation

#### 13.300.1.1 test\_BruhatGenerator()

```
template<class Field, FFLAS_DIAG diag, class RandIter >
bool test_BruhatGenerator (
 const Field & F,
 size_t n,
 size_t r,
 size_t t,
 typename Field::ConstElement_ptr A,
 size_t lda,
 typename Field::Element_ptr TS,
 size_t l,
 RandIter & G)
```

#### 13.300.1.2 launch\_test()

```
template<class Field, FFLAS_DIAG diag, class RandIter >
bool launch_test (
 const Field & F,
 size_t n,
 size_t r,
 size_t t,
 size_t l,
 RandIter & G)
```

#### 13.300.1.3 testLTQSRPM()

```
template<class Field, class RandGen >
bool testLTQSRPM (
 const Field & F,
```

```

 size_t n,
 size_t r,
 size_t t,
 RandGen & G)

```

#### 13.300.1.4 run\_with\_field()

```

template<class Field >
bool run_with_field (
 Givaro::Integer q,
 uint64_t b,
 size_t n,
 size_t r,
 size_t t,
 size_t l,
 size_t iters,
 uint64_t seed)

```

#### 13.300.1.5 main()

```

int main (
 int argc,
 char ** argv)

```

## 13.301 test-rankprofiles.C File Reference

```

#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/ffpack/ffpack.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include <givaro/modular.h>
#include <iostream>
#include <iomanip>
#include <random>
#include <chrono>

```

### Macros

- `#define __FFLASFFPACK_SEQUENTIAL`

### Functions

- `template<class Field >`  
`bool run_with_field (Givaro::Integer q, uint64_t b, size_t m, size_t n, size_t r, size_t iters, uint64_t seed, bool par)`
- `int main (int argc, char **argv)`

## 13.301.1 Macro Definition Documentation

### 13.301.1.1 \_\_FFLASFFPACK\_SEQUENTIAL

```
#define __FFLASFFPACK_SEQUENTIAL
```

## 13.301.2 Function Documentation

### 13.301.2.1 run\_with\_field()

```
template<class Field >
```

```
bool run_with_field (
 Givaro::Integer q,
 uint64_t b,
 size_t m,
 size_t n,
 size_t r,
 size_t iters,
 uint64_t seed,
 bool par)
```

### 13.301.2.2 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.302 test-rpm.C File Reference

```
#include <iostream>
#include "givaro/modular.h"
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/ffpack/ffpack.h"
```

### Functions

- bool [checkRPM](#) (size\_t M, size\_t N, size\_t R)
- bool [checkSymmetricRPM](#) (size\_t N, size\_t R)
- int [main](#) (int argc, char \*\*argv)

### 13.302.1 Function Documentation

#### 13.302.1.1 checkRPM()

```
bool checkRPM (
 size_t M,
 size_t N,
 size_t R)
```

#### 13.302.1.2 checkSymmetricRPM()

```
bool checkSymmetricRPM (
 size_t N,
 size_t R)
```

#### 13.302.1.3 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.303 test-simd.C File Reference

```
#include "givaro/givinteger.h"
#include "givaro/modular.h"
#include "fflas-ffpack/fflas-ffpack-config.h"
```

```
#include "fflas-ffpack/fflas/fflas_simd.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/utils/align-allocator.h"
#include <array>
#include <vector>
#include <random>
#include <string>
#include <functional>
#include <limits>
#include <type_traits>
#include <algorithm>
```

## Data Structures

- struct [ALL< true, v... >](#)
- struct [ALL< false, v... >](#)
- struct [ALL<>](#)
- struct [count\\_nonconst\\_lvalue\\_reference< T, O... >](#)
- struct [count\\_nonconst\\_lvalue\\_reference< T &, O... >](#)
- struct [count\\_nonconst\\_lvalue\\_reference< const T &, O... >](#)
- struct [count\\_nonconst\\_lvalue\\_reference<>](#)
- struct [is\\_all\\_same< T, Args... >](#)
- struct [is\\_all\\_same<>](#)
- struct [width< T >](#)
- struct [width< float >](#)
- struct [width< double >](#)
- class [TestOneMethod< Simd >](#)
- struct [ScalFunctionsBase< Element, typename enable\\_if< is\\_floating\\_point< Element >::value >::type >](#)
- class [ScalFunctionsBase< Element, typename enable\\_if< is\\_floating\\_point< Element >::value >::type >::FloatingPointTestD](#)
- struct [ScalFunctionsBase< Element, typename enable\\_if< is\\_integral< Element >::value >::type >](#)
- struct [ScalFunctions< Element >](#)

## Macros

- [#define \\_TEST\\_ONE\(K, f1, f2, r, n\)](#)
- [#define TEST\\_ONE\\_OP\(f\)](#)
- [#define TEST\\_ONE\\_OP\\_WZ\(f\)](#)
- [#define TEST\\_IMPL\(SIZE, Elt\)](#)

## Functions

- [template<typename Element > enable\\_if< is\\_integral< Element >::value, bool >::type \[check\\\_eq\]\(#\) \(Element x, Element y\)](#)
- [template<typename Element > enable\\_if< is\\_floating\\_point< Element >::value, bool >::type \[check\\\_eq\]\(#\) \(Element x, Element y\)](#)
- [template<typename Element > bool \[cmp\]\(#\) \(vector< Element > out\\_scal, vector< Element > out\\_simd\)](#)
- [template<typename Ret , typename T > Ret \[eval\\\_func\\\_on\\\_array\]\(#\) \(function< Ret\(\)> f, array< T, 0 > &arr\)](#)
- [template<typename T , typename... TArgs> void \[eval\\\_func\\\_on\\\_array\]\(#\) \(function< void\(T, TArgs...\)> f, array< typename decay< T >::type, sizeof...\(TArgs\)+1 > &arr\)](#)
- [template<typename Ret , typename T , typename... TArgs> Ret \[eval\\\_func\\\_on\\\_array\]\(#\) \(function< Ret\(T, TArgs...\)> f, array< typename decay< T >::type, sizeof...\(TArgs\)+1 > &arr\)](#)



- `template<typename E >`  
`std::ostream & operator<< (std::ostream &o, const vector< E > &V)`
- `template<typename Simd, typename Element >`  
`enable_if< is_floating_point< Element >::value, bool >::type test_impl_base ()`
- `template<typename Simd, typename Element >`  
`enable_if< is_integral< Element >::value, bool >::type test_impl_base ()`
- `template<typename Simd, typename Element >`  
`bool test_impl ()`
- `int main (int argc, char *argv[])`

## 13.303.1 Macro Definition Documentation

### 13.303.1.1 \_TEST\_ONE

```
#define _TEST_ONE(
```

```
 K,
 f1,
 f2,
 r,
 n)
```

Value:

```
do { \
 K T(f1, f2, r, n); \
 bool b = T.writeResultLine(); \
 if (b == false) \
 T.writeDebugData(); \
 btest &= b; \
} while (0)
```

### 13.303.1.2 TEST\_ONE\_OP

```
#define TEST_ONE_OP(
```

```
 f)
```

Value:

```
_TEST_ONE(TestOneMethod<Simd>, \
function<decltype(Simd::f)>(Simd::f), \
function<decltype(Scal::f)>(Scal::f), \
function<decltype(Scal::genInputs)>(Scal::genInputs), #f)
```

### 13.303.1.3 TEST\_ONE\_OP\_WZ

```
#define TEST_ONE_OP_WZ(
```

```
 f)
```

Value:

```
_TEST_ONE(TestOneMethod<Simd>, \
function<decltype(Simd::f)>(Simd::f), \
function<decltype(Scal::f)>(Scal::f), \
function<decltype(Scal::genInputsWithZero)>(Scal::genInputsWithZero), \
#f " test with zero")
```

### 13.303.1.4 TEST\_IMPL

```
#define TEST_IMPL(
```

```
 SIZE,
 Elt)
```

Value:

```
do { \
 pass &= test_impl<Simd##SIZE<Elt>, Elt>(); \
 cout << endl; \
} while (0)
```

## 13.303.2 Function Documentation

### 13.303.2.1 check\_eq() [1/2]

```
template<typename Element >
```

```
enable_if< is_integral< Element >::value, bool >::type check_eq (
 Element x,
 Element y)
```

#### 13.303.2.2 check\_eq() [2/2]

```
template<typename Element >
enable_if< is_floating_point< Element >::value, bool >::type check_eq (
 Element x,
 Element y)
```

#### 13.303.2.3 cmp()

```
template<typename Element >
bool cmp (
 vector< Element > out_scal,
 vector< Element > out_simd)
```

#### 13.303.2.4 eval\_func\_on\_array() [1/3]

```
template<typename Ret , typename T >
Ret eval_func_on_array (
 function< Ret ()> f,
 array< T, 0 > & arr)
```

#### 13.303.2.5 eval\_func\_on\_array() [2/3]

```
template<typename T , typename... TArgs>
void eval_func_on_array (
 function< void(T, TArgs...)> f,
 array< typename decay< T >::type, sizeof...(TArgs)+1 > & arr)
```

#### 13.303.2.6 eval\_func\_on\_array() [3/3]

```
template<typename Ret , typename T , typename... TArgs>
Ret eval_func_on_array (
 function< Ret(T, TArgs...)> f,
 array< typename decay< T >::type, sizeof...(TArgs)+1 > & arr)
```

#### 13.303.2.7 operator<<()

```
template<typename E >
std::ostream & operator<< (
 std::ostream & o,
 const vector< E > & V)
```

#### 13.303.2.8 test\_impl\_base() [1/2]

```
template<typename Simd , typename Element >
enable_if< is_floating_point< Element >::value, bool >::type test_impl_base ()
```

#### 13.303.2.9 test\_impl\_base() [2/2]

```
template<typename Simd , typename Element >
enable_if< is_integral< Element >::value, bool >::type test_impl_base ()
```

#### 13.303.2.10 test\_impl()

```
template<typename Simd , typename Element >
bool test_impl ()
```

**13.303.2.11 main()**

```
int main (
 int argc,
 char * argv[])
```

**13.304 test-solve.C File Reference**

```
#include <givaro/modular-integer.h>
#include <iomanip>
#include <iostream>
#include "fflas-ffpack/utils/timer.h"
#include "fflas-ffpack/fflas/fflas.h"
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include <givaro/modular.h>
#include <givaro/modular-balanced.h>
```

**Functions**

- `template<typename Field , class RandIter >`  
`bool check_solve (const Field &F, size_t m, RandIter &Rand, bool isParallel)`
- `template<class Field >`  
`bool run_with_field (Givaro::Integer q, size_t b, size_t m, size_t iters, uint64_t seed)`
- `int main (int argc, char **argv)`

**13.304.1 Function Documentation****13.304.1.1 check\_solve()**

```
template<typename Field , class RandIter >
bool check_solve (
 const Field & F,
 size_t m,
 RandIter & Rand,
 bool isParallel)
```

**13.304.1.2 run\_with\_field()**

```
template<class Field >
bool run_with_field (
 Givaro::Integer q,
 size_t b,
 size_t m,
 size_t iters,
 uint64_t seed)
```

**13.304.1.3 main()**

```
int main (
 int argc,
 char ** argv)
```

**13.305 test-storage-transpose.C File Reference**

```
#include <iomanip>
#include <iostream>
```

```
#include <random>
#include "fflas-ffpack/utils/args-parser.h"
#include "fflas-ffpack/utils/test-utils.h"
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/fflas_memory.h"
#include <givaro/modular.h>
#include <recint/rint.h>
#include "fflas-ffpack/fflas/fflas_transpose.h"
```

## Data Structures

- class [Test< Elt >](#)

## Functions

- [int main \(int argc, char \\*\\*argv\)](#)

### 13.305.1 Function Documentation

#### 13.305.1.1 main()

```
int main (
 int argc,
 char ** argv)
```

### 13.306 101-fgemm.C File Reference

```
#include <fflas-ffpack/fflas-ffpack-config.h>
#include <givaro/modular-balanced.h>
#include <fflas-ffpack/fflas/fflas.h>
#include <fflas-ffpack/utils/timer.h>
#include <fflas-ffpack/utils/fflas_io.h>
#include <fflas-ffpack/utils/args-parser.h>
#include <iostream>
```

## Functions

- [int main \(int argc, char \\*\\*argv\)](#)

### 13.306.1 Function Documentation

#### 13.306.1.1 main()

```
int main (
 int argc,
 char ** argv)
```

### 13.307 2x2-fgemm.C File Reference

```
#include <fflas-ffpack/fflas-ffpack-config.h>
#include <givaro/modular-balanced.h>
#include <fflas-ffpack/fflas/fflas.h>
#include <fflas-ffpack/utils/timer.h>
#include <fflas-ffpack/utils/fflas_io.h>
#include <fflas-ffpack/utils/args-parser.h>
```

```
#include <iostream>
```

## Functions

- `int main (int argc, char **argv)`

### 13.307.1 Function Documentation

#### 13.307.1.1 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.308 2x2-fftrsv.C File Reference

```
#include <fflas-ffpack/fflas-ffpack-config.h>
#include <givaro/modular-balanced.h>
#include <fflas-ffpack/fflas/fflas.h>
#include <fflas-ffpack/utils/timer.h>
#include <fflas-ffpack/utils/fflas_io.h>
#include <fflas-ffpack/utils/args-parser.h>
#include <iostream>
#include <array>
```

## Functions

- `int main (int argc, char **argv)`

### 13.308.1 Function Documentation

#### 13.308.1.1 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.309 2x2-pluq.C File Reference

```
#include <iostream>
#include <vector>
#include <givaro/modular.h>
#include "fflas-ffpack/fflas-ffpack-config.h"
#include "fflas-ffpack/fflas-ffpack.h"
#include "fflas-ffpack/utils/fflas_io.h"
```

## Functions

- `int main (int argc, char **argv)`

### 13.309.1 Function Documentation

#### 13.309.1.1 main()

```
int main (
```

```
int argc,
char ** argv)
```

## 13.310 fflas-101\_1.C File Reference

```
#include <fflas-ffpack/fflas/fflas.h>
#include <givaro/modular.h>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/utils/fflas_io.h"
#include <iostream>
```

### Functions

- `int main (int argc, char **argv)`

### 13.310.1 Function Documentation

#### 13.310.1.1 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.311 fflas-101\_3.C File Reference

```
#include <fflas-ffpack/fflas/fflas.h>
#include <givaro/modular.h>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/utils/fflas_io.h"
#include "fflas-ffpack/utils/fflas_randommatrix.h"
#include <iostream>
```

### Functions

- `int main (int argc, char **argv)`

### 13.311.1 Function Documentation

#### 13.311.1.1 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.312 fflas\_101.C File Reference

```
#include <fflas-ffpack/fflas/fflas.h>
#include <givaro/modular.h>
#include <givaro/modular-balanced.h>
#include <iostream>
```

### Functions

- `int main (int argc, char **argv)`

## 13.312.1 Function Documentation

### 13.312.1.1 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.313 fflas\_101\_lvl1.C File Reference

```
#include <fflas-ffpack/fflas/fflas.h>
#include <givaro/modular.h>
#include <givaro/modular-balanced.h>
#include <iostream>
#include "fflas-ffpack/utils/fflas_io.h"
```

### Functions

- `int main (int argc, char **argv)`

## 13.313.1 Function Documentation

### 13.313.1.1 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.314 ffpack-fgesv.C File Reference

```
#include <fflas-ffpack/fflas/fflas.h>
#include <givaro/modular.h>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/utils/fflas_io.h"
#include <fflas-ffpack/ffpack/ffpack.h>
#include <iostream>
```

### Functions

- `int main (int argc, char **argv)`

## 13.314.1 Function Documentation

### 13.314.1.1 main()

```
int main (
 int argc,
 char ** argv)
```

## 13.315 ffpack-solve.C File Reference

```
#include <fflas-ffpack/fflas/fflas.h>
#include <givaro/modular.h>
#include <givaro/modular-balanced.h>
#include "fflas-ffpack/utils/fflas_io.h"
#include <fflas-ffpack/ffpack/ffpack.h>
```

```
#include <iostream>
```

## Functions

- `int main (int argc, char **argv)`

### 13.315.1 Function Documentation

#### 13.315.1.1 main()

```
int main (
 int argc,
 char ** argv)
```

PS: the function Solve will modify the matrix A so here we used a duplicate matrix A2 otherwise  $A \cdot x$  will not be equal to b for the later verification stage



# Index

- [\\_F](#)
  - [RNSIntegerMod< RNS >, 549](#)
- [\\_M](#)
  - [rns\\_double, 528](#)
  - [rns\\_double\\_extended, 539](#)
- [\\_MAX\\_SIZE\\_MATRICES](#)
  - [benchmark-checkers.C, 777](#)
- [\\_MMi](#)
  - [rns\\_double, 528](#)
  - [rns\\_double\\_extended, 539](#)
- [\\_Mi](#)
  - [rns\\_double, 528](#)
  - [rns\\_double\\_extended, 539](#)
- [\\_Mi\\_modp\\_rns](#)
  - [RNSIntegerMod< RNS >, 549](#)
- [\\_NR\\_TESTS](#)
  - [benchmark-checkers.C, 777](#)
- [\\_PLUQ](#)
  - [FFPACK, 368](#)
- [\\_RNSdelayed](#)
  - [RNSIntegerMod< RNS >, 549](#)
- [\\_TEST\\_ONE](#)
  - [test-simd.C, 1099](#)
- [\\_\\_FFLASFFPACK\\_ARITHPROG\\_THRESHOLD](#)
  - [fflas-ffpack-default-thresholds.h, 824](#)
- [\\_\\_FFLASFFPACK\\_CACHE\\_LINE\\_SIZE](#)
  - [fflas\\_sparse.h, 884](#)
- [\\_\\_FFLASFFPACK\\_CHARPOLY\\_Danilevskii\\_LUKrylov\\_THRESHOLD](#)
  - [fflas-ffpack-default-thresholds.h, 824](#)
- [\\_\\_FFLASFFPACK\\_CHARPOLY\\_LUKrylov\\_ArithProg\\_THRESHOLD](#)
  - [fflas-ffpack-default-thresholds.h, 824](#)
- [\\_\\_FFLASFFPACK\\_CONFIGURATION](#)
  - [cblas.C, 1048](#)
  - [clapack.C, 1049](#)
  - [fblas.C, 1050](#)
  - [lapack.C, 1050](#)
- [\\_\\_FFLASFFPACK\\_DIMKPENALTY](#)
  - [fflas\\_pfgemm.inl, 869](#)
- [\\_\\_FFLASFFPACK\\_FORCE\\_SEQ](#)
  - [benchmark-charpoly-mp.C, 776](#)
- [\\_\\_FFLASFFPACK\\_FSYRK\\_THRESHOLD](#)
  - [fflas-ffpack-default-thresholds.h, 824](#)
- [\\_\\_FFLASFFPACK\\_FSYTRF\\_THRESHOLD](#)
  - [fflas-ffpack-default-thresholds.h, 824](#)
- [\\_\\_FFLASFFPACK\\_FTRSSYR2K\\_THRESHOLD](#)
  - [ffpack.h, 922](#)
- [\\_\\_FFLASFFPACK\\_FTRSTR\\_THRESHOLD](#)
  - [ffpack.h, 922](#)
- [\\_\\_FFLASFFPACK\\_FTRTRI\\_THRESHOLD](#)
  - [fflas-ffpack-default-thresholds.h, 824](#)
- [\\_\\_FFLASFFPACK\\_GAUSSJORDAN\\_BASECASE](#)
  - [ffpack\\_echelonforms.inl, 930](#)
  - [test-echelon.C, 1056](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_BIG\\_ENDIAN](#)
  - [config.h, 805](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_BLAS](#)
  - [config.h, 805](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_CBLAS](#)
  - [cblas.C, 1048](#)
  - [config.h, 805](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_CLAPACK](#)
  - [clapack.C, 1049](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_CXX11](#)
  - [config.h, 805](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_DGETRF](#)
  - [benchmark-dgetrf.C, 779](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_DLFCN\\_H](#)
  - [config.h, 805](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_DTRTRI](#)
  - [benchmark-dtrtri.C, 782](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_FLOAT\\_H](#)
  - [config.h, 805](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_INT128](#)
  - [config.h, 805](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_INTTYPES\\_H](#)
  - [config.h, 805](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_LAPACK](#)
  - [clapack.C, 1049](#)
  - [config.h, 805](#)
  - [lapack.C, 1050](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_LIMITS\\_H](#)
  - [config.h, 805](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_PTHREAD\\_H](#)
  - [config.h, 805](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_STDDEF\\_H](#)
  - [config.h, 806](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_STDINT\\_H](#)
  - [config.h, 806](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_STDIO\\_H](#)
  - [config.h, 806](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_STDLIB\\_H](#)
  - [config.h, 806](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_STRINGS\\_H](#)
  - [config.h, 806](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_STRING\\_H](#)
  - [config.h, 806](#)
- [\\_\\_FFLASFFPACK\\_HAVE\\_SYS\\_STAT\\_H](#)
  - [config.h, 806](#)

- \_\_FFLASFFPACK\_HAVE\_SYS\_TIME\_H  
config.h, [806](#)
- \_\_FFLASFFPACK\_HAVE\_SYS\_TYPES\_H  
config.h, [806](#)
- \_\_FFLASFFPACK\_HAVE\_UNISTD\_H  
config.h, [806](#)
- \_\_FFLASFFPACK\_KAAPI\_ROUTINES\_INL  
kaapi\_routines.inl, [1030](#)
- \_\_FFLASFFPACK\_LT\_OBJDIR  
config.h, [806](#)
- \_\_FFLASFFPACK\_MINBLOCKCUTS  
blockcuts.inl, [1029](#)
- \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET  
benchmark-charpoly.C, [776](#)  
benchmark-fadd-lvl2.C, [782](#)  
benchmark-fdot.C, [783](#)  
benchmark-fgemm-mp.C, [784](#)  
benchmark-fgemm-rns.C, [785](#)  
benchmark-fgemv-mp.C, [787](#)  
benchmark-fgemv.C, [789](#)  
benchmark-fgesv.C, [791](#)  
benchmark-fsyrc.C, [792](#)  
benchmark-fsytrf.C, [793](#)  
benchmark-ftrsm-mp.C, [794](#)  
benchmark-ftrsm.C, [794](#)  
benchmark-ftrsv.C, [795](#)  
benchmark-ftrtri.C, [795](#)  
benchmark-pluq.C, [798](#)  
benchmark-quasisep.C, [799](#)
- \_\_FFLASFFPACK\_OPENBLAS\_NUM\_THREADS  
config.h, [806](#)
- \_\_FFLASFFPACK\_PACKAGE  
config.h, [806](#)
- \_\_FFLASFFPACK\_PACKAGE\_BUGREPORT  
config.h, [806](#)
- \_\_FFLASFFPACK\_PACKAGE\_NAME  
config.h, [806](#)
- \_\_FFLASFFPACK\_PACKAGE\_STRING  
config.h, [807](#)
- \_\_FFLASFFPACK\_PACKAGE\_TARNAME  
config.h, [807](#)
- \_\_FFLASFFPACK\_PACKAGE\_URL  
config.h, [807](#)
- \_\_FFLASFFPACK\_PACKAGE\_VERSION  
config.h, [807](#)
- \_\_FFLASFFPACK\_PLUQ\_THRESHOLD  
fflas-ffpack-default-thresholds.h, [824](#)  
test-echelon.C, [1056](#)
- \_\_FFLASFFPACK\_SEQPARTHRESHOLD  
fflas\_pfgemm.inl, [869](#)
- \_\_FFLASFFPACK\_SEQUENTIAL  
parallel.h, [1031](#)  
test-echelon.C, [1056](#)  
test-ftrmm.C, [1076](#)  
test-ftrmv.C, [1077](#)  
test-ftrsm.C, [1079](#)  
test-ftrsv.C, [1082](#)  
test-ftrtri.C, [1083](#)  
test-lu.C, [1086](#)  
test-rankprofiles.C, [1096](#)
- \_\_FFLASFFPACK\_SIZEOF\_CHAR  
config.h, [807](#)
- \_\_FFLASFFPACK\_SIZEOF\_INT  
config.h, [807](#)
- \_\_FFLASFFPACK\_SIZEOF\_LONG  
config.h, [807](#)
- \_\_FFLASFFPACK\_SIZEOF\_LONG\_LONG  
config.h, [807](#)
- \_\_FFLASFFPACK\_SIZEOF\_SHORT  
config.h, [807](#)
- \_\_FFLASFFPACK\_SIZEOF\_\_\_INT64\_T  
config.h, [807](#)
- \_\_FFLASFFPACK\_STDC\_HEADERS  
config.h, [807](#)
- \_\_FFLASFFPACK\_USE\_OPENMP  
config.h, [807](#)
- \_\_FFLASFFPACK\_VERSION  
config.h, [807](#)
- \_\_FFLASFFPACK\_WINOTHRESHOLD  
fflas-ffpack-default-thresholds.h, [823](#)
- \_\_FFLASFFPACK\_WINOTHRESHOLD\_BAL  
fflas-ffpack-default-thresholds.h, [823](#)
- \_\_FFLASFFPACK\_WINOTHRESHOLD\_BAL\_FLT  
fflas-ffpack-default-thresholds.h, [824](#)
- \_\_FFLASFFPACK\_WINOTHRESHOLD\_FLT  
fflas-ffpack-default-thresholds.h, [823](#)
- \_\_FFLASFFPACK\_charpoly\_INL  
ffpack\_charpoly.inl, [925](#)
- \_\_FFLASFFPACK\_checker\_charpoly\_INL  
checker\_charpoly.inl, [812](#)
- \_\_FFLASFFPACK\_checker\_det\_INL  
checker\_det.inl, [812](#)
- \_\_FFLASFFPACK\_checker\_fgemm\_INL  
checker\_fgemm.inl, [813](#)
- \_\_FFLASFFPACK\_checker\_ftrsm\_INL  
checker\_ftrsm.inl, [813](#)
- \_\_FFLASFFPACK\_checker\_invert\_INL  
checker\_invert.inl, [814](#)
- \_\_FFLASFFPACK\_checker\_pluq\_INL  
checker\_pluq.inl, [814](#)
- \_\_FFLASFFPACK\_fadd\_INL  
fflas\_fadd.inl, [830](#)
- \_\_FFLASFFPACK\_fassign\_INL  
fflas\_fassign.inl, [831](#)
- \_\_FFLASFFPACK\_faxpy\_INL  
fflas\_faxpy.inl, [832](#)
- \_\_FFLASFFPACK\_fdot\_INL  
fflas\_fdot.inl, [833](#)
- \_\_FFLASFFPACK\_fflas\_blockcuts\_INL  
blockcuts.inl, [1029](#)
- \_\_FFLASFFPACK\_fflas\_bounds\_INL  
fflas\_bounds.inl, [827](#)
- \_\_FFLASFFPACK\_fflas\_fflas\_fgemm\_winograd\_INL  
fgemm\_winograd.inl, [838](#)
- \_\_FFLASFFPACK\_fflas\_fflas\_level1\_INL  
fflas\_level1.inl, [864](#)

- \_\_FFLASFFPACK\_fflas\_fflas\_level2\_INL  
fflas\_level2.inl, [866](#)
- \_\_FFLASFFPACK\_fflas\_fflas\_level3\_INL  
fflas\_level3.inl, [869](#)
- \_\_FFLASFFPACK\_fflas\_fflas\_mmhelper\_INL  
fflas\_helpers.inl, [858](#)
- \_\_FFLASFFPACK\_fflas\_fflas\_sparse\_INL  
fflas\_sparse.inl, [886](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_INL  
simd128.inl, [872](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_double\_INL  
simd128\_double.inl, [873](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_float\_INL  
simd128\_float.inl, [873](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_int16\_INL  
simd128\_int16.inl, [874](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_int32\_INL  
simd128\_int32.inl, [874](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_int64\_INL  
simd128\_int64.inl, [875](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_INL  
simd256.inl, [875](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_double\_INL  
simd256\_double.inl, [876](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_float\_INL  
simd256\_float.inl, [876](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_int16\_INL  
simd256\_int16.inl, [877](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_int32\_INL  
simd256\_int32.inl, [877](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_int64\_INL  
simd256\_int64.inl, [877](#)
- \_\_FFLASFFPACK\_fflas\_freduce\_INL  
fflas\_freduce.inl, [848](#)
- \_\_FFLASFFPACK\_fflas\_freduce\_mp\_INL  
fflas\_freduce\_mp.inl, [848](#)
- \_\_FFLASFFPACK\_fflas\_fsy2k\_INL  
fflas\_fsy2k.inl, [852](#)
- \_\_FFLASFFPACK\_fflas\_fsyrk\_INL  
fflas\_fsyrk.inl, [854](#)
- \_\_FFLASFFPACK\_fflas\_fsyrk\_strassen\_INL  
fflas\_fsyrk\_strassen.inl, [855](#)
- \_\_FFLASFFPACK\_fflas\_igemm\_igemm\_INL  
igemm.inl, [859](#)
- \_\_FFLASFFPACK\_fflas\_igemm\_igemm\_kernels\_INL  
igemm\_kernels.inl, [861](#)
- \_\_FFLASFFPACK\_fflas\_igemm\_igemm\_tools\_INL  
igemm\_tools.inl, [861](#)
- \_\_FFLASFFPACK\_fflas\_pfgemm\_INL  
fflas\_pfgemm.inl, [869](#)
- \_\_FFLASFFPACK\_fflas\_pftsrsm\_INL  
fflas\_pftsrsm.inl, [870](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_pspmm\_INL  
csr\_hyb\_pspmm.inl, [895](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_pspmv\_INL  
csr\_hyb\_pspmv.inl, [895](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_spm INL  
csr\_hyb\_spm.inl, [896](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_spmv\_INL  
csr\_hyb\_spmv.inl, [896](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_utils\_INL  
csr\_hyb\_utils.inl, [897](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_pspmm\_INL  
csr\_pspmm.inl, [890](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_pspmv\_INL  
csr\_pspmv.inl, [891](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_spm INL  
csr\_spm.inl, [892](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_spmv\_INL  
csr\_spmv.inl, [893](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_pspmm\_INL  
ell\_pspmm.inl, [898](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_pspmv\_INL  
ell\_pspmv.inl, [899](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_simd\_pspmv\_INL  
ell\_simd\_pspmv.inl, [902](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_simd\_spmv\_INL  
ell\_simd\_spmv.inl, [903](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_simd\_utils\_INL  
ell\_simd\_utils.inl, [904](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_spm INL  
ell\_spm.inl, [900](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_spmv\_INL  
ell\_spmv.inl, [901](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_utils\_INL  
ell\_utils.inl, [901](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_pspmm\_INL  
hyb\_zo\_pspmm.inl, [905](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_pspmv\_INL  
hyb\_zo\_pspmv.inl, [905](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_spm INL  
hyb\_zo\_spm.inl, [906](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_spmv\_INL  
hyb\_zo\_spmv.inl, [906](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_utils\_INL  
hyb\_zo\_utils.inl, [906](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_coo\_spm INL  
coo\_spm.inl, [888](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_coo\_spmv\_INL  
coo\_spmv.inl, [889](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_coo\_utils\_INL  
coo\_utils.inl, [889](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_sell\_pspmv\_INL  
sell\_pspmv.inl, [908](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_sell\_spmv\_INL  
sell\_spmv.inl, [909](#)
- \_\_FFLASFFPACK\_fflas\_sparse\_sell\_utils\_INL  
sell\_utils.inl, [910](#)
- \_\_FFLASFFPACK\_ffpack\_INL  
ffpack.inl, [923](#)
- \_\_FFLASFFPACK\_ffpack\_bruhatgen\_inl  
ffpack\_bruhatgen.inl, [925](#)
- \_\_FFLASFFPACK\_ffpack\_charpoly\_danilveski\_INL  
ffpack\_charpoly\_danilevski.inl, [926](#)
- \_\_FFLASFFPACK\_ffpack\_charpoly\_kgfast\_INL  
ffpack\_charpoly\_kgfast.inl, [926](#)

\_\_FFLASFFPACK\_ffpack\_charpoly\_kgfastgeneralized\_INL  
     ffpack\_charpoly\_kgfastgeneralized.inl, 927  
 \_\_FFLASFFPACK\_ffpack\_charpoly\_kglu\_INL  
     ffpack\_charpoly\_kglu.inl, 927  
 \_\_FFLASFFPACK\_ffpack\_echelon\_forms\_INL  
     ffpack\_echelonforms.inl, 930  
 \_\_FFLASFFPACK\_ffpack\_fgesv\_INL  
     ffpack\_fgesv.inl, 930  
 \_\_FFLASFFPACK\_ffpack\_fgetrs\_INL  
     ffpack\_fgetrs.inl, 931  
 \_\_FFLASFFPACK\_ffpack\_fsytrf\_INL  
     ffpack\_fsytrf.inl, 933  
 \_\_FFLASFFPACK\_ffpack\_ftrssyr2k\_INL  
     ffpack\_ftrssyr2k.inl, 933  
 \_\_FFLASFFPACK\_ffpack\_ftrstr\_INL  
     ffpack\_ftrstr.inl, 934  
 \_\_FFLASFFPACK\_ffpack\_ftrtr\_INL  
     ffpack\_ftrtr.inl, 934  
 \_\_FFLASFFPACK\_ffpack\_invert\_INL  
     ffpack\_invert.inl, 935  
 \_\_FFLASFFPACK\_ffpack\_krylovelim\_INL  
     ffpack\_krylovelim.inl, 935  
 \_\_FFLASFFPACK\_ffpack\_ludivine\_INL  
     ffpack\_ludivine.inl, 936  
 \_\_FFLASFFPACK\_ffpack\_minpoly\_INL  
     ffpack\_minpoly.inl, 937  
 \_\_FFLASFFPACK\_ffpack\_permutation\_INL  
     ffpack\_permutation.inl, 939  
 \_\_FFLASFFPACK\_ffpack\_pluq\_INL  
     ffpack\_pluq.inl, 940  
 \_\_FFLASFFPACK\_ffpack\_ppluq\_INL  
     ffpack\_ppluq.inl, 941  
 \_\_FFLASFFPACK\_ffpack\_rank\_profiles\_INL  
     ffpack\_rankprofiles.inl, 943  
 \_\_FFLASFFPACK\_ffgemm\_INL  
     fflas\_fgemm.inl, 835  
 \_\_FFLASFFPACK\_ffgemm\_bini\_INL  
     schedule\_bini.inl, 839  
 \_\_FFLASFFPACK\_ffgemm\_winograd\_INL  
     schedule\_winograd.inl, 839  
 \_\_FFLASFFPACK\_ffgemm\_winograd\_acc\_INL  
     schedule\_winograd\_acc.inl, 840  
 \_\_FFLASFFPACK\_ffgemm\_winograd\_acc\_ip\_INL  
     schedule\_winograd\_acc\_ip.inl, 841  
 \_\_FFLASFFPACK\_ffgemm\_winograd\_ip\_INL  
     schedule\_winograd\_ip.inl, 841  
 \_\_FFLASFFPACK\_ffgemv\_INL  
     fflas\_ffgemv.inl, 843  
 \_\_FFLASFFPACK\_ffgemv\_mp\_INL  
     fflas\_ffgemv\_mp.inl, 844  
 \_\_FFLASFFPACK\_ffger\_INL  
     fflas\_ffger.inl, 845  
 \_\_FFLASFFPACK\_field\_rns\_INL  
     rns.inl, 949  
 \_\_FFLASFFPACK\_field\_rns\_double\_INL  
     rns-double.inl, 947  
 \_\_FFLASFFPACK\_field\_rns\_double\_recint\_INL  
     rns-double-recint.inl, 946  
 \_\_FFLASFFPACK\_freivalds\_INL  
     fflas\_freivalds.inl, 849  
 \_\_FFLASFFPACK\_fscal\_INL  
     fflas\_fscal.inl, 850  
 \_\_FFLASFFPACK\_fscal\_mp\_INL  
     fflas\_fscal\_mp.inl, 851  
 \_\_FFLASFFPACK\_ftrmm\_INL  
     fflas\_ftrmm.inl, 855  
 \_\_FFLASFFPACK\_ftrsm\_INL  
     fflas\_ftrsm.inl, 856  
 \_\_FFLASFFPACK\_ftrsv\_INL  
     fflas\_ftrsv.inl, 857  
 \_\_FFLASFFPACK\_simd512\_INL  
     simd512.inl, 878  
 \_\_FFLASFFPACK\_simd512\_double\_INL  
     simd512\_double.inl, 878  
 \_\_FFLASFFPACK\_simd512\_float\_INL  
     simd512\_float.inl, 879  
 \_\_FFLASFFPACK\_simd512\_int32\_INL  
     simd512\_int32.inl, 879  
 \_\_FFLAS\_L1\_INST\_C  
     fflas\_L1\_inst.C, 961  
 \_\_FFLAS\_L2\_INST\_C  
     fflas\_L2\_inst.C, 965  
 \_\_FFLAS\_L3\_INST\_C  
     fflas\_L3\_inst.C, 968  
 \_\_FFLAS\_\_TRSM\_READONLY  
     fflas\_L3\_inst\_implement.inl, 971  
     fflas\_level3.inl, 869  
     ffpack\_ppluq.inl, 941  
 \_\_FFPACK\_FSYTRF\_BC\_CROUT  
     benchmark-fsytrf.C, 793  
 \_\_FFPACK\_INST\_C  
     ffpack\_inst.C, 1023  
 \_\_FFPACK\_charpoly\_mp\_INL  
     ffpack\_charpoly\_mp.inl, 928  
 \_\_FFPACK\_det\_mp\_INL  
     ffpack\_det\_mp.inl, 928  
 \_\_FFPACK\_ffgemm\_classical\_INL  
     ffgemm\_classical\_mp.inl, 837  
 \_\_FFPACK\_fger\_mp\_INL  
     fflas\_fger\_mp.inl, 845  
 \_\_FFPACK\_ftrsm\_mp\_INL  
     fflas\_ftrsm\_mp.inl, 856  
 \_\_FFPACK\_ludivine\_mp\_INL  
     ffpack\_ludivine\_mp.inl, 936  
 \_\_FFPACK\_pluq\_mp\_INL  
     ffpack\_pluq\_mp.inl, 941  
 \_\_LUDIVINE\_CUTOFF  
     test-lu.C, 1087  
 \_\_has\_builtin  
     bit\_manipulation.h, 1040  
 \_\_alloc  
     rns\_double\_elt, 530  
     rns\_double\_elt\_cstptr, 533  
     rns\_double\_elt\_ptr, 535  
 \_\_basis  
     rns\_double, 528

- rns\_double\_extended, 538
- \_basisMax
  - rns\_double, 528
  - rns\_double\_extended, 538
- \_coo
  - SpMat< Field, flag >, 754
- \_coo16
  - CooMat< Field >, 439
- \_coo16\_zo
  - CooMat< Field >, 439
- \_coo32
  - CooMat< Field >, 439
- \_coo32\_zo
  - CooMat< Field >, 439
- \_coo64
  - CooMat< Field >, 439
- \_coo64\_zo
  - CooMat< Field >, 439
- \_crt\_in
  - rns\_double, 528
  - rns\_double\_extended, 539
- \_crt\_out
  - rns\_double, 528
  - rns\_double\_extended, 539
- \_csr
  - SpMat< Field, flag >, 754
- \_csr16
  - CsrMat< Field >, 441
- \_csr16\_zo
  - CsrMat< Field >, 441
- \_csr32
  - CsrMat< Field >, 441
- \_csr32\_zo
  - CsrMat< Field >, 441
- \_csr64
  - CsrMat< Field >, 441
- \_csr64\_zo
  - CsrMat< Field >, 441
- \_ell
  - SpMat< Field, flag >, 754
- \_ell16
  - EllMat< Field >, 447
- \_ell16\_zo
  - EllMat< Field >, 447
- \_ell32
  - EllMat< Field >, 447
- \_ell32\_zo
  - EllMat< Field >, 447
- \_ell64
  - EllMat< Field >, 447
- \_ell64\_zo
  - EllMat< Field >, 447
- \_errorStream
  - Failure, 449
- \_field\_rns
  - rns\_double, 528
  - rns\_double\_extended, 539
- \_iM\_modp\_rns
  - RNSIntegerMod< RNS >, 549
- \_invbasis
  - rns\_double, 528
  - rns\_double\_extended, 538
- \_ldm
  - rns\_double, 529
  - rns\_double\_extended, 539
- \_mi\_sum
  - rns\_double, 529
- \_mm
  - Test< Elt >, 762
- \_negbasis
  - rns\_double, 528
  - rns\_double\_extended, 538
- \_nn
  - Test< Elt >, 762
- \_p
  - RNSIntegerMod< RNS >, 549
- \_pbits
  - rns\_double, 529
  - rns\_double\_extended, 539
- \_ptr
  - rns\_double\_elt, 530
  - rns\_double\_elt\_cstptr, 533
  - rns\_double\_elt\_ptr, 535
- \_rns
  - RNSInteger< RNS >, 543
  - RNSIntegerMod< RNS >, 549
- \_simd512\_int64\_INL
  - simd512\_int64.inl, 880
- \_size
  - rns\_double, 528
  - rns\_double\_extended, 539
- \_stride
  - rns\_double\_elt, 530
  - rns\_double\_elt\_cstptr, 533
  - rns\_double\_elt\_ptr, 535
- \_zero
  - ScalFunctionsBase< Element, typename enable\_if< is\_floating\_point< Element >::value >::type >, 558
  - ScalFunctionsBase< Element, typename enable\_if< is\_integral< Element >::value >::type >, 560
- ~CheckerImplem\_Det
  - CheckerImplem\_Det< Field >, 419
- ~CheckerImplem\_PLUQ
  - CheckerImplem\_PLUQ< Field >, 424
- ~CheckerImplem\_charpoly
  - CheckerImplem\_charpoly< Field, Polynomial >, 417
  - CheckerImplem\_charpoly< Givaro::ZRing< Givaro::Integer >, Polynomial >, 418
- ~CheckerImplem\_fgmm
  - CheckerImplem\_fgmm< Field >, 421
- ~CheckerImplem\_ftsm
  - CheckerImplem\_ftsm< Field >, 422
- ~CheckerImplem\_invert
  - CheckerImplem\_invert< Field >, 423

- ~rns\_double\_elt
  - rns\_double\_elt, 529
- 101-fgemm.C, 1102
  - main, 1102
- 2x2-fgemm.C, 1102
  - main, 1103
- 2x2-ftsrv.C, 1103
  - main, 1103
- 2x2-pluq.C, 1103
  - main, 1103
- add
  - FFLAS::vectorised, 285
  - FieldSimd< \_Field >, 452
  - RNSIntegerMod< RNS >, 547
  - ScalFunctions< Element >, 553
  - Simd128\_impl< true, true, false, 2 >, 569
  - Simd128\_impl< true, true, false, 4 >, 579
  - Simd128\_impl< true, true, false, 8 >, 590
  - Simd128\_impl< true, true, true, 2 >, 597
  - Simd128\_impl< true, true, true, 4 >, 606
  - Simd128\_impl< true, true, true, 8 >, 615
  - Simd256\_impl< true, false, true, 8 >, 626
  - Simd256\_impl< true, true, false, 2 >, 637
  - Simd256\_impl< true, true, false, 4 >, 653
  - Simd256\_impl< true, true, false, 8 >, 665
  - Simd256\_impl< true, true, true, 2 >, 673
  - Simd256\_impl< true, true, true, 4 >, 684, 690
  - Simd256\_impl< true, true, true, 8 >, 699
  - Simd512\_impl< true, false, true, 8 >, 708
  - Simd512\_impl< true, true, false, 8 >, 719
  - Simd512\_impl< true, true, true, 8 >, 727
- add\_r
  - FieldSimd< \_Field >, 452, 453
- addin
  - FieldSimd< \_Field >, 452
  - ScalFunctions< Element >, 553
  - Simd128\_impl< true, true, false, 2 >, 569
  - Simd128\_impl< true, true, false, 4 >, 579
  - Simd128\_impl< true, true, false, 8 >, 590
  - Simd128\_impl< true, true, true, 2 >, 598
  - Simd128\_impl< true, true, true, 4 >, 606
  - Simd128\_impl< true, true, true, 8 >, 616
  - Simd256\_impl< true, false, true, 8 >, 626
  - Simd256\_impl< true, true, false, 2 >, 637
  - Simd256\_impl< true, true, false, 4 >, 654
  - Simd256\_impl< true, true, false, 8 >, 665
  - Simd256\_impl< true, true, true, 2 >, 673
  - Simd256\_impl< true, true, true, 4 >, 684, 690
  - Simd256\_impl< true, true, true, 8 >, 699
  - Simd512\_impl< true, false, true, 8 >, 708
  - Simd512\_impl< true, true, false, 8 >, 719
  - Simd512\_impl< true, true, true, 8 >, 727
- addin\_r
  - FieldSimd< \_Field >, 453
- addp
  - FFLAS::vectorised, 284
- AlgoChooser< ModeCategories::ConvertTo< Element-  
Categories::RNSElementTag >, ParSeq >, 405
  - value, 405
- AlgoChooser< ModeT, ParSeq >, 405
  - value, 405
- align-allocator.h, 1038
- alignable
  - FFLAS, 191
- alignable< Givaro::Integer \* >
  - FFLAS, 192
- aligned\_allocator
  - NoSimd< T >, 518
  - Simd128\_impl< true, true, false, 2 >, 564
  - Simd128\_impl< true, true, false, 4 >, 574
  - Simd128\_impl< true, true, false, 8 >, 584
  - Simd128\_impl< true, true, true, 2 >, 594
  - Simd128\_impl< true, true, true, 4 >, 603
  - Simd128\_impl< true, true, true, 8 >, 613
  - Simd256\_impl< true, false, true, 8 >, 623
  - Simd256\_impl< true, true, false, 2 >, 631
  - Simd256\_impl< true, true, false, 4 >, 643
  - Simd256\_impl< true, true, false, 8 >, 660
  - Simd256\_impl< true, true, true, 2 >, 670
  - Simd256\_impl< true, true, true, 4 >, 680
  - Simd256\_impl< true, true, true, 8 >, 696
  - Simd512\_impl< true, false, true, 8 >, 705
  - Simd512\_impl< true, true, false, 8 >, 713
  - Simd512\_impl< true, true, true, 8 >, 724
- aligned\_vector
  - NoSimd< T >, 518
  - Simd128\_impl< true, true, false, 2 >, 564
  - Simd128\_impl< true, true, false, 4 >, 574
  - Simd128\_impl< true, true, false, 8 >, 584
  - Simd128\_impl< true, true, true, 2 >, 595
  - Simd128\_impl< true, true, true, 4 >, 604
  - Simd128\_impl< true, true, true, 8 >, 613
  - Simd256\_impl< true, false, true, 8 >, 623
  - Simd256\_impl< true, true, false, 2 >, 631
  - Simd256\_impl< true, true, false, 4 >, 643
  - Simd256\_impl< true, true, false, 8 >, 660
  - Simd256\_impl< true, true, true, 2 >, 670
  - Simd256\_impl< true, true, true, 4 >, 680, 681
  - Simd256\_impl< true, true, true, 8 >, 696
  - Simd512\_impl< true, false, true, 8 >, 705
  - Simd512\_impl< true, true, false, 8 >, 713
  - Simd512\_impl< true, true, true, 8 >, 724
- alignment
  - FieldSimd< \_Field >, 456
  - NoSimd< T >, 519
  - Simd128\_impl< true, true, false, 2 >, 572
  - Simd128\_impl< true, true, false, 4 >, 582
  - Simd128\_impl< true, true, false, 8 >, 592
  - Simd128\_impl< true, true, true, 2 >, 601
  - Simd128\_impl< true, true, true, 4 >, 610
  - Simd128\_impl< true, true, true, 8 >, 620
  - Simd256\_impl< true, false, true, 8 >, 629
  - Simd256\_impl< true, true, false, 2 >, 639



- Simd256\_impl< true, true, false, 4 >, [657](#)
- Simd256\_impl< true, true, false, 8 >, [667](#)
- Simd256\_impl< true, true, true, 2 >, [676](#)
- Simd256\_impl< true, true, true, 4 >, [693](#)
- Simd256\_impl< true, true, true, 8 >, [702](#)
- Simd512\_impl< true, false, true, 8 >, [710](#)
- Simd512\_impl< true, true, false, 8 >, [721](#)
- Simd512\_impl< true, true, true, 8 >, [731](#)
- ALL< false, v... >, [406](#)
  - value, [406](#)
- ALL< true, v... >, [406](#)
  - value, [406](#)
- ALL< v >, [405](#)
- ALL<>, [406](#)
  - value, [406](#)
- Amax
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [500](#)
- Amin
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [500](#)
- applyP
  - FFPACK, [309](#), [310](#), [370](#)
- applyP\_block
  - FFPACK, [362](#)
- applyP\_modular\_double
  - ffpack.C, [988](#)
  - ffpack\_c.h, [1010](#)
- ArbitraryPrecIntTag, [406](#)
- Architecture of the library., [2](#)
- areEqual
  - RNSIntegerMod< RNS >, [548](#)
- AreEqual< X, X >, [407](#)
  - value, [407](#)
- AreEqual< X, Y >, [407](#)
  - value, [407](#)
- args-parser.h, [1038](#)
  - ArgumentType, [1039](#)
  - END\_OF\_ARGUMENTS, [1039](#)
  - findArgument, [1039](#)
  - getListArgs, [1039](#)
  - printHelpMessage, [1039](#)
  - TYPE\_BOOL, [1039](#)
  - TYPE\_DOUBLE, [1039](#)
  - TYPE\_INT, [1039](#)
  - TYPE\_INTEGER, [1039](#)
  - type\_integer, [1039](#)
  - TYPE\_INTLIST, [1039](#)
  - TYPE\_LONGLONG, [1039](#)
  - TYPE\_NONE, [1039](#)
  - TYPE\_STR, [1039](#)
  - TYPE\_UINT64, [1039](#)
- Argument, [407](#)
  - c, [407](#)
  - data, [408](#)
  - example, [407](#)
  - helpString, [407](#)
  - type, [408](#)
- ArgumentType
  - args-parser.h, [1039](#)
- ArithProg
  - FFPACK::Protected, [400](#)
- arithprog.C, [771](#)
  - CUBE, [771](#)
  - GFOPS, [771](#)
  - main, [772](#)
  - TTimer, [771](#)
- assign
  - RNSInteger< RNS >, [542](#)
  - RNSIntegerMod< RNS >, [547](#)
- associatedDelayedField< const FFPACK::RNSIntegerMod< RNS > >, [408](#)
  - field, [408](#)
  - type, [408](#)
- associatedDelayedField< const Givaro::Modular< T, X > >, [409](#)
  - field, [409](#)
  - type, [409](#)
- associatedDelayedField< const Givaro::ModularBalanced< T > >, [409](#)
  - field, [409](#)
  - type, [409](#)
- associatedDelayedField< const Givaro::ZRing< T > >, [410](#)
  - field, [410](#)
  - type, [410](#)
- associatedDelayedField< Field >, [408](#)
  - field, [408](#)
  - type, [408](#)
- assume\_aligned
  - fflas\_sparse.h, [884](#)
- AtlasConj
  - config-blas.h, [818](#)
- Aunfit
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [498](#)
- aut
  - HelperFlag, [472](#)
- Auto, [410](#)
- averageCol
  - StatsMatrix, [756](#)
- averageColDifference
  - StatsMatrix, [756](#)
- averageRow
  - StatsMatrix, [756](#)
- averageRowDifference
  - StatsMatrix, [756](#)
- axpy
  - FieldSimd< \_Field >, [454](#), [455](#)
- axpy\_r
  - FieldSimd< \_Field >, [455](#)
- axpyin
  - FieldSimd< \_Field >, [455](#)
  - RNSIntegerMod< RNS >, [547](#)
- axpyin\_r
  - FieldSimd< \_Field >, [455](#)

- axpyp
  - FFLAS::vectorised, [285](#)
  - FFLAS::vectorised::unswitch, [289](#)
- balanced
  - FieldTraits< FFPACK::RNSInteger< T > >, [457](#)
  - FieldTraits< FFPACK::RNSIntegerMod< T > >, [458](#)
  - FieldTraits< Field >, [457](#)
  - FieldTraits< Givaro::Modular< Element > >, [458](#)
  - FieldTraits< Givaro::ModularBalanced< Element > >, [459](#)
  - FieldTraits< Givaro::ZRing< double > >, [459](#)
  - FieldTraits< Givaro::ZRing< float > >, [460](#)
  - FieldTraits< Givaro::ZRing< Givaro::Integer > >, [460](#)
  - FieldTraits< Givaro::ZRing< int16\_t > >, [461](#)
  - FieldTraits< Givaro::ZRing< int32\_t > >, [461](#)
  - FieldTraits< Givaro::ZRing< int64\_t > >, [461](#)
  - FieldTraits< Givaro::ZRing< Reclnt::ruint< K > >, [462](#)
  - FieldTraits< Givaro::ZRing< uint16\_t > >, [462](#)
  - FieldTraits< Givaro::ZRing< uint32\_t > >, [463](#)
  - FieldTraits< Givaro::ZRing< uint64\_t > >, [463](#)
  - winograd.C, [775](#)
- BARRIER
  - parallel.h, [1031](#)
- BASECASE\_K
  - test-lu.C, [1086](#)
- BaseTimer
  - FFLAS, [74](#)
- BasisElement
  - rns\_double, [525](#)
  - rns\_double\_extended, [536](#)
  - RNSInteger< RNS >, [540](#)
  - RNSIntegerMod< RNS >, [545](#)
- begin
  - Info, [476](#), [477](#)
- BEGIN\_PARALLEL\_MAIN
  - parallel.h, [1032](#)
- Bench
  - Bench< Elt >, [412](#)
- Bench< Elt >, [410](#)
  - Bench, [412](#)
  - cardinality, [412](#)
  - doBenchs, [412](#)
  - Elt\_ptr, [411](#)
  - enable\_if\_no\_simd\_t, [411](#)
  - enable\_if\_simd128\_t, [411](#)
  - enable\_if\_simd256\_t, [411](#)
  - enable\_if\_simd512\_t, [411](#)
  - enable\_if\_t, [411](#)
  - F, [412](#)
  - Field, [411](#)
  - inplace, [413](#)
  - is\_same\_element, [411](#)
  - iters, [413](#)
  - m, [412](#)
  - n, [412](#)
  - Residu, [411](#)
  - run, [412](#)
- benchmark-charpoly-mp.C, [775](#)
  - \_\_FFLASFFPACK\_FORCE\_SEQ, [776](#)
  - main, [776](#)
- benchmark-charpoly.C, [776](#)
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, [776](#)
  - main, [777](#)
  - run\_with\_field, [776](#)
- benchmark-checkers.C, [777](#)
  - \_MAX\_SIZE\_MATRICES, [777](#)
  - \_NR\_TESTS, [777](#)
  - CUBE, [777](#)
  - ENABLE\_ALL\_CHECKINGS, [777](#)
  - main, [778](#)
- benchmark-dgemm.C, [778](#)
  - CBLAS\_GEMM, [778](#)
  - Floats, [778](#)
  - main, [778](#)
  - TTimer, [778](#)
- benchmark-dgetrf.C, [779](#)
  - \_\_FFLASFFPACK\_HAVE\_DGETRF, [779](#)
  - main, [779](#)
  - TTimer, [779](#)
- benchmark-dgetri.C, [779](#)
  - main, [780](#)
  - TTimer, [780](#)
- benchmark-dsytrf.C, [780](#)
  - EFFGFF, [780](#)
  - main, [781](#)
  - TTimer, [780](#)
- benchmark-dtrsm.C, [781](#)
  - main, [781](#)
  - TTimer, [781](#)
- benchmark-dtrtri.C, [781](#)
  - \_\_FFLASFFPACK\_HAVE\_DTRTRI, [782](#)
  - main, [782](#)
  - TTimer, [782](#)
- benchmark-fadd-lvl2.C, [782](#)
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, [782](#)
  - main, [782](#)
- benchmark-fdot.C, [783](#)
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, [783](#)
  - main, [783](#)
  - run\_with\_field, [783](#)
- benchmark-fgemm-mp.C, [783](#)
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, [784](#)
  - main, [784](#)
  - MG\_DEFAULT, [784](#)
  - STD\_RECINT\_SIZE, [784](#)
  - tmain, [784](#)
- benchmark-fgemm-rns.C, [784](#)
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, [785](#)



- ConstElement\_ptr, 785
- Element\_ptr, 785
- Field, 785
- GRAIN, 785
- main, 786
- PSeq, 786
- RNS, 785
- THREADS, 785
- THREED, 785
- THREEDA, 786
- THREEDIP, 786
- TWOD, 785
- TWODA, 785
- benchmark-fgemm.C, 786
  - CLASSIC\_HYBRID, 786
  - main, 786
- benchmark-fgemv-mp.C, 787
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, 787
  - main, 787
  - MG\_DEFAULT, 787
  - STD\_RECINT\_SIZE, 787
  - tmain, 787
  - write\_matrix, 787
- benchmark-fgemv.C, 788
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, 789
  - benchmark\_disp, 790
  - benchmark\_in\_Field, 790
  - benchmark\_with\_field, 790
  - benchmark\_with\_timer, 789
  - check\_result, 789
  - fill\_value, 789
  - genData, 789
  - main, 791
- benchmark-fgesv.C, 791
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, 791
  - main, 791
- benchmark-fsyr2k.C, 791
  - main, 792
- benchmark-fsyrk.C, 792
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, 792
  - main, 792
- benchmark-fsytrf.C, 792
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, 793
  - \_\_FFPACK\_FSYTRF\_BC\_CROUT, 793
  - CUBE, 793
  - main, 793
- benchmark-ftdsm-mp.C, 793
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, 794
  - main, 794
- benchmark-ftdsm.C, 794
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, 794
- main, 794
- benchmark-ftsv.C, 794
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, 795
  - main, 795
- benchmark-fttri.C, 795
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, 795
  - CUBE, 795
  - main, 795
- benchmark-inverse.C, 796
  - CUBE, 796
  - main, 796
- benchmark-lqup-mp.C, 796
  - main, 796
- benchmark-lqup.C, 797
  - CUBE, 797
  - main, 797
- benchmark-pluq.C, 797
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, 798
  - CUBE, 798
  - Field, 798
  - main, 798
  - Rec\_Initialize, 798
  - verification\_PLUQ, 798
- benchmark-quasiseq.C, 798
  - \_\_FFLASFFPACK\_OPENBLAS\_NT\_ALREADY\_SET, 799
  - main, 799
  - run\_with\_field, 799
- benchmark-storage-transpose.C, 799
  - main, 800
- benchmark-wino.C, 800
  - CUBE, 800
  - launch\_wino, 800
  - main, 800
- benchmark\_disp
  - benchmark-fgemv.C, 790
- benchmark\_in\_Field
  - benchmark-fgemv.C, 790
- benchmark\_with\_field
  - benchmark-fgemv.C, 790
- benchmark\_with\_timer
  - benchmark-fgemv.C, 789
- Bibliography, 7
- Bini, 413
- FFLAS::BLAS3, 196
- bit\_manipulation.h, 1040
  - \_\_has\_builtin, 1040
  - clz, 1040
  - ctz, 1040
- bitsize
  - FFLAS, 142
- bitsize< Givaro::ZRing< Givaro::Integer > >
  - FFLAS, 142
- blas\_enum
  - config-blas.h, 817

- blend
  - ScalFunctions< Element >, 556
  - Simd128\_impl< true, true, false, 2 >, 569
  - Simd128\_impl< true, true, false, 4 >, 579
  - Simd128\_impl< true, true, false, 8 >, 589
  - Simd128\_impl< true, true, true, 2 >, 597
  - Simd128\_impl< true, true, true, 4 >, 606
  - Simd128\_impl< true, true, true, 8 >, 615
  - Simd256\_impl< true, false, true, 8 >, 626
  - Simd256\_impl< true, true, false, 2 >, 637
  - Simd256\_impl< true, true, false, 4 >, 653
  - Simd256\_impl< true, true, false, 8 >, 665
  - Simd256\_impl< true, true, true, 2 >, 673
  - Simd256\_impl< true, true, true, 4 >, 683, 690
  - Simd256\_impl< true, true, true, 8 >, 699
  - Simd512\_impl< true, false, true, 8 >, 708
  - Simd512\_impl< true, true, false, 8 >, 719
  - Simd512\_impl< true, true, true, 8 >, 727
- blendv
  - ScalFunctionsBase< Element, typename enable\_if< is\_floating\_point< Element >::value >::type >, 557
  - Simd256\_impl< true, false, true, 8 >, 626
  - Simd512\_impl< true, false, true, 8 >, 708
- Block, 413
- BlockCuts
  - FFLAS, 181, 183
- BlockCuts< CuttingStrategy::Block, StrategyParameter::Fixed >
  - FFLAS, 183
- BlockCuts< CuttingStrategy::Block, StrategyParameter::Grain >
  - FFLAS, 182
- BlockCuts< CuttingStrategy::Block, StrategyParameter::Threads >
  - FFLAS, 183
- BlockCuts< CuttingStrategy::Column, StrategyParameter::Fixed >
  - FFLAS, 182
- BlockCuts< CuttingStrategy::Column, StrategyParameter::Grain >
  - FFLAS, 182
- BlockCuts< CuttingStrategy::Column, StrategyParameter::Threads >
  - FFLAS, 183
- BlockCuts< CuttingStrategy::Row, StrategyParameter::Fixed >
  - FFLAS, 182
- BlockCuts< CuttingStrategy::Row, StrategyParameter::Grain >
  - FFLAS, 182
- BlockCuts< CuttingStrategy::Row, StrategyParameter::Threads >
  - FFLAS, 183
- BlockCuts< CuttingStrategy::Single, StrategyParameter::Threads >
  - FFLAS, 181
- blockcuts.inl, 1028
  - \_\_FFLASFFPACK\_MINBLOCKCUTS, 1029
  - \_\_FFLASFFPACK\_fflas\_blockcuts\_INL, 1029
- BlockingFactor
  - FFLAS::details, 210
- BlockTransposeSIMD< Field, Simd, >, 413
  - info, 413
  - size, 413
  - transpose, 414
- Bmax
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait >, 500
- Bmin
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait >, 500
- Bruhat2EchelonPermutation
  - FFPACK, 350
- Bug List, 3
- buildMatrix
  - FFPACK, 356
- Bunfit
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait >, 498
- c
  - Argument, 407
- callLUdivine\_small< double >, 415
  - operator(), 415
- callLUdivine\_small< Element >, 415
  - operator(), 415
- callLUdivine\_small< float >, 416
  - operator(), 416
- cardinality
  - Bench< Elt >, 412
  - RNSInteger< RNS >, 542
  - RNSIntegerMod< RNS >, 546
  - Test< Elt >, 761
- cast.h, 1041
- category
  - FieldTraits< FFPACK::RNSInteger< T > >, 457
  - FieldTraits< FFPACK::RNSIntegerMod< T > >, 457
  - FieldTraits< Field >, 456
  - FieldTraits< Givaro::Modular< Element > >, 458
  - FieldTraits< Givaro::ModularBalanced< Element > >, 458
  - FieldTraits< Givaro::ZRing< double > >, 459
  - FieldTraits< Givaro::ZRing< float > >, 459
  - FieldTraits< Givaro::ZRing< Givaro::Integer > >, 460
  - FieldTraits< Givaro::ZRing< int16\_t > >, 460
  - FieldTraits< Givaro::ZRing< int32\_t > >, 461
  - FieldTraits< Givaro::ZRing< int64\_t > >, 461
  - FieldTraits< Givaro::ZRing< Reclnt::ruint< K > >, 462
  - FieldTraits< Givaro::ZRing< uint16\_t > >, 462
  - FieldTraits< Givaro::ZRing< uint32\_t > >, 463
  - FieldTraits< Givaro::ZRing< uint64\_t > >, 463
- cblas.C, 1048
  - \_\_FFLASFFPACK\_CONFIGURATION, 1048

- \_\_FFLASFFPACK\_HAVE\_CBLAS, 1048
- main, 1048
- CBLAS\_DIAG
  - config-blas.h, 818
- cblas\_dsyrc
  - config-blas.h, 822
- CBLAS\_ENUM\_DEFINED\_H
  - config-blas.h, 817
- CBLAS\_EXTERNALS
  - config-blas.h, 817
- CBLAS\_GEMM
  - benchmark-dgemm.C, 778
- cblas\_imprsm
  - FFLAS, 130
- CBLAS\_INT
  - config-blas.h, 817
- CBLAS\_ORDER
  - config-blas.h, 817
- CBLAS\_SIDE
  - config-blas.h, 818
- CBLAS\_TRANSPOSE
  - config-blas.h, 818
- CBLAS\_UPLO
  - config-blas.h, 818
- CblasColMajor
  - config-blas.h, 817
- CblasConjTrans
  - config-blas.h, 818
- CblasLeft
  - config-blas.h, 818
- CblasLower
  - config-blas.h, 818
- CblasNonUnit
  - config-blas.h, 818
- CblasNoTrans
  - config-blas.h, 818
- CblasRight
  - config-blas.h, 818
- CblasRowMajor
  - config-blas.h, 817
- CblasTrans
  - config-blas.h, 818
- CblasUnit
  - config-blas.h, 818
- CblasUpper
  - config-blas.h, 818
- ceil
  - ScalFunctionsBase< Element, typename enable\_if< is\_floating\_point< Element >::value >::type >, 557
  - Simd256\_impl< true, false, true, 8 >, 628
  - Simd512\_impl< true, false, true, 8 >, 710
- characteristic
  - RNSInteger< RNS >, 541
  - RNSIntegerMod< RNS >, 546
- CharPoly
  - FFPACK, 327, 328, 356, 374, 375
- charpoly.C, 808
- CUBE, 808
- GFOPS, 808
  - main, 808, 809
- TTimer, 808
- CharpolyFailed, 416
- check
  - Checker\_Empty< Field >, 417
  - CheckerImplem\_charpoly< Field, Polynomial >, 418
  - CheckerImplem\_charpoly< Givaro::ZRing< Givaro::Integer >, Polynomial >, 419
  - CheckerImplem\_Det< Field >, 419
  - CheckerImplem\_fgemm< Field >, 422
  - CheckerImplem\_ftrsm< Field >, 423
  - CheckerImplem\_invert< Field >, 423
  - CheckerImplem\_PLUQ< Field >, 424
- check1
  - regression-check.C, 1051
- check2
  - regression-check.C, 1051
- check3
  - regression-check.C, 1051
- check4
  - regression-check.C, 1051
- check\_computeS1S2
  - test-fsyrc.C, 1073
- CHECK\_DEPENDENCIES
  - parallel.h, 1031
- check\_eq
  - test-simd.C, 1099, 1100
- check\_fdot
  - test-fdot.C, 1060
- check\_fger
  - test-fger.C, 1066
- check\_fsyr2k
  - test-fsyr2k.C, 1071
- check\_fsyrk
  - test-fsyrc.C, 1073
- check\_fsyrk\_bkdiag
  - test-fsyrc.C, 1073
- check\_fsyrk\_diag
  - test-fsyrc.C, 1073
- check\_ftrmm
  - test-ftrmm.C, 1076
- check\_ftrmv
  - test-ftrmv.C, 1077
- check\_ftrsm
  - test-ftrsm.C, 1079
- check\_ftrssyr2k
  - test-ftrssyr2k.C, 1080
- check\_ftrstr
  - test-ftrstr.C, 1081
- check\_ftrsv
  - test-ftrsv.C, 1082
- check\_ftrtri
  - test-ftrtri.C, 1083
- check\_minpoly
  - test-minpoly.C, 1091

- check\_MM
  - test-fgemm.C, 1062
- check\_MV
  - test-fgemv.C, 1064
- check\_result
  - benchmark-fgemv.C, 789
- check\_solve
  - test-solve.C, 1101
- checkA
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, 498
- checkB
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, 499
- CHECKER, 41
- Checker\_charpoly
  - FFPACK, 308
- checker\_charpoly.inl, 812
  - \_\_FFLASFFPACK\_checker\_charpoly\_INL, 812
- Checker\_Det
  - FFPACK, 307
- checker\_det.inl, 812
  - \_\_FFLASFFPACK\_checker\_det\_INL, 812
- Checker\_Empty
  - Checker\_Empty< Field >, 417
- Checker\_Empty< Field >, 416
  - check, 417
  - Checker\_Empty, 417
- checker\_empty.h, 812
- Checker\_fgemm
  - FFLAS, 72
- checker\_fgemm.inl, 813
  - \_\_FFLASFFPACK\_checker\_fgemm\_INL, 813
- Checker\_ftsm
  - FFLAS, 72
- checker\_ftsm.inl, 813
  - \_\_FFLASFFPACK\_checker\_ftsm\_INL, 813
- Checker\_invert
  - FFPACK, 308
- checker\_invert.inl, 813
  - \_\_FFLASFFPACK\_checker\_invert\_INL, 814
- Checker\_PLUQ
  - FFPACK, 307
- checker\_pluq.inl, 814
  - \_\_FFLASFFPACK\_checker\_pluq\_INL, 814
- CheckerImplem\_charpoly
  - CheckerImplem\_charpoly< Field, Polynomial >, 417
  - CheckerImplem\_charpoly< Givaro::ZRing< Givaro::Integer >, Polynomial >, 418
- CheckerImplem\_charpoly< Field, Polynomial >, 417
  - ~CheckerImplem\_charpoly, 417
  - check, 418
  - CheckerImplem\_charpoly, 417
- CheckerImplem\_charpoly< Givaro::ZRing< Givaro::Integer >, Polynomial >, 418
  - ~CheckerImplem\_charpoly, 418
  - check, 419
- CheckerImplem\_charpoly, 418
  - Ring, 418
- CheckerImplem\_Det
  - CheckerImplem\_Det< Field >, 419
- CheckerImplem\_Det< Field >, 419
  - ~CheckerImplem\_Det, 419
  - check, 419
  - CheckerImplem\_Det, 419
- CheckerImplem\_fgemm
  - CheckerImplem\_fgemm< Field >, 421
- CheckerImplem\_fgemm< Field >, 421
  - ~CheckerImplem\_fgemm, 421
  - check, 422
  - CheckerImplem\_fgemm, 421
- CheckerImplem\_ftsm
  - CheckerImplem\_ftsm< Field >, 422
- CheckerImplem\_ftsm< Field >, 422
  - ~CheckerImplem\_ftsm, 422
  - check, 423
  - CheckerImplem\_ftsm, 422
- CheckerImplem\_invert
  - CheckerImplem\_invert< Field >, 423
- CheckerImplem\_invert< Field >, 423
  - ~CheckerImplem\_invert, 423
  - check, 423
  - CheckerImplem\_invert, 423
- CheckerImplem\_PLUQ
  - CheckerImplem\_PLUQ< Field >, 424
- CheckerImplem\_PLUQ< Field >, 424
  - ~CheckerImplem\_PLUQ, 424
  - check, 424
  - CheckerImplem\_PLUQ, 424
- checkers.doxy, 814
- checkers\_fflas.h, 814
- checkers\_fflas.inl, 815
  - FFLASFFPACK\_checkers\_fflas\_inl\_H, 815
- checkers\_ffpack.h, 815
- checkers\_ffpack.inl, 816
  - FFLASFFPACK\_checkers\_ffpack\_inl\_H, 816
- checkingMessage
  - test-nullspace.C, 1092
- checkMonotonicApplyP
  - test-permutations.C, 1093
- checkOut
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, 499
- checkRPM
  - test-rpm.C, 1097
- checkSymmetricRPM
  - test-rpm.C, 1097
- checkZeroDimCharpoly
  - regression-check.C, 1051
- checkZeroDimMinPoly
  - regression-check.C, 1051
- chooseField
  - FFPACK, 395
- chooseField< Givaro::ZRing< double > >
  - FFPACK, 395

- chooseField< Givaro::ZRing< float > >  
FFPACK, [395](#)
- chooseField< Givaro::ZRing< int32\_t > >  
FFPACK, [395](#)
- chooseField< Givaro::ZRing< int64\_t > >  
FFPACK, [395](#)
- chunk
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd >, [744](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd\_ZO >, [746](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL >, [750](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, [753](#)
- chunkSize
  - Sparse< \_Field, SparseMatrix\_t::SELL >, [751](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, [754](#)
- clapack.C, [1048](#)
  - \_\_FFLASFFPACK\_CONFIGURATION, [1049](#)
  - \_\_FFLASFFPACK\_HAVE\_CLAPACK, [1049](#)
  - \_\_FFLASFFPACK\_HAVE\_LAPACK, [1049](#)
  - main, [1049](#)
- Classic, [425](#)
- CLASSIC\_HYBRID
  - benchmark-fgemm.C, [786](#)
- clz
  - bit\_manipulation.h, [1040](#)
- Cmax
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [500](#)
- Cmin
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [500](#)
- cmp
  - test-simd.C, [1100](#)
- cmp\_false
  - ScalFunctionsBase< Element, typename enable\_if< is\_floating\_point< Element >::value >::type >, [558](#)
  - ScalFunctionsBase< Element, typename enable\_if< is\_integral< Element >::value >::type >, [561](#)
- cmp\_true
  - ScalFunctionsBase< Element, typename enable\_if< is\_floating\_point< Element >::value >::type >, [558](#)
  - ScalFunctionsBase< Element, typename enable\_if< is\_integral< Element >::value >::type >, [560](#)
- col
  - Coo< Field >, [436](#)
  - Coo< ValT, IdxT >, [435](#), [438](#)
  - Sparse< \_Field, SparseMatrix\_t::COO >, [734](#)
  - Sparse< \_Field, SparseMatrix\_t::COO\_ZO >, [736](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR >, [738](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR\_HYB >, [739](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR\_ZO >, [741](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL >, [743](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd >, [745](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd\_ZO >, [746](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_ZO >, [748](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL >, [751](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, [754](#)
- coldim
  - StatsMatrix, [755](#)
- ColRankProfileSubmatrix
  - FFPACK, [340](#), [379](#)
- ColRankProfileSubmatrix\_modular\_double
  - ffpack.C, [1000](#)
  - ffpack\_c.h, [1020](#)
- ColRankProfileSubmatrixIndices
  - FFPACK, [339](#), [379](#)
- ColRankProfileSubmatrixIndices\_modular\_double
  - ffpack.C, [1000](#)
  - ffpack\_c.h, [1020](#)
- Column, [425](#)
- ColumnEchelonForm
  - FFPACK, [321](#), [373](#)
- ColumnEchelonForm\_modular\_double
  - ffpack.C, [991](#)
  - ffpack\_c.h, [1013](#)
- ColumnEchelonForm\_modular\_float
  - ffpack.C, [991](#)
  - ffpack\_c.h, [1013](#)
- ColumnEchelonForm\_modular\_int32\_t
  - ffpack.C, [992](#)
  - ffpack\_c.h, [1013](#)
- ColumnRankProfile
  - FFPACK, [336](#), [337](#), [378](#)
- ColumnRankProfile\_modular\_double
  - ffpack.C, [999](#)
  - ffpack\_c.h, [1019](#)
- COMMA
  - parallel.h, [1034](#)
- CompactElement< double >, [425](#)
  - type, [425](#)
- CompactElement< Element >, [425](#)
  - type, [425](#)
- CompactElement< float >, [426](#)
  - type, [426](#)
- CompactElement< int16\_t >, [426](#)
  - type, [426](#)
- CompactElement< int32\_t >, [426](#)
  - type, [426](#)
- CompactElement< int64\_t >, [426](#)
  - type, [426](#)
- compatible\_data\_type< Field >, [427](#)
  - value, [427](#)
- compatible\_data\_type< Givaro::ZRing< double > >, [427](#)
  - value, [427](#)
- compatible\_data\_type< Givaro::ZRing< float > >, [427](#)
  - value, [427](#)
- compliant
  - NoSimd< T >, [519](#)
  - Simd128\_impl< true, true, false, 2 >, [567](#)
  - Simd128\_impl< true, true, false, 4 >, [577](#)
  - Simd128\_impl< true, true, false, 8 >, [587](#)

- Simd128\_impl< true, true, true, 2 >, [595](#)
- Simd128\_impl< true, true, true, 4 >, [604](#)
- Simd128\_impl< true, true, true, 8 >, [613](#)
- Simd256\_impl< true, false, true, 8 >, [624](#)
- Simd256\_impl< true, true, false, 2 >, [634](#)
- Simd256\_impl< true, true, false, 4 >, [649](#)
- Simd256\_impl< true, true, false, 8 >, [663](#)
- Simd256\_impl< true, true, true, 2 >, [670](#)
- Simd256\_impl< true, true, true, 4 >, [681](#), [687](#)
- Simd256\_impl< true, true, true, 8 >, [696](#)
- Simd512\_impl< true, false, true, 8 >, [705](#)
- Simd512\_impl< true, true, false, 8 >, [716](#)
- Simd512\_impl< true, true, true, 8 >, [724](#)
- Compose
  - Compose< H1, H2 >, [428](#)
- Compose< H1, H2 >, [427](#)
  - Compose, [428](#)
  - first\_component, [428](#)
  - operator<<, [428](#)
  - second\_component, [428](#)
- composePermutationsLLL
  - FFPACK, [365](#)
  - ffpack.C, [987](#)
  - ffpack\_c.h, [1009](#)
- composePermutationsLLM
  - FFPACK, [365](#)
  - ffpack.C, [987](#)
  - ffpack\_c.h, [1009](#)
- composePermutationsMLM
  - FFPACK, [366](#)
  - ffpack.C, [987](#)
  - ffpack\_c.h, [1009](#)
- CompressRows
  - FFPACK::Protected, [401](#)
- CompressRowsQA
  - FFPACK::Protected, [402](#)
- CompressRowsQK
  - FFPACK::Protected, [402](#)
- CompressToBlockBiDiagonal
  - FFPACK, [349](#)
- computeDeviation
  - FFLAS, [155](#)
- computeFactorClassic
  - FFLAS::Protected, [216](#)
- ComputeRPermutation
  - FFPACK, [351](#), [354](#)
- computeS1S2
  - FFLAS, [125](#)
- config-blas.h, [816](#)
  - AtlasConj, [818](#)
  - blas\_enum, [817](#)
  - CBLAS\_DIAG, [818](#)
  - cblas\_dsyrk, [822](#)
  - CBLAS\_ENUM\_DEFINED\_H, [817](#)
  - CBLAS\_EXTERNALS, [817](#)
  - CBLAS\_INT, [817](#)
  - CBLAS\_ORDER, [817](#)
  - CBLAS\_SIDE, [818](#)
  - CBLAS\_TRANSPOSE, [818](#)
  - CBLAS\_UPLO, [818](#)
  - CblasColMajor, [817](#)
  - CblasConjTrans, [818](#)
  - CblasLeft, [818](#)
  - CblasLower, [818](#)
  - CblasNonUnit, [818](#)
  - CblasNoTrans, [818](#)
  - CblasRight, [818](#)
  - CblasRowMajor, [817](#)
  - CblasTrans, [818](#)
  - CblasUnit, [818](#)
  - CblasUpper, [818](#)
  - dasum\_, [819](#)
  - daxpy\_, [818](#)
  - dcopy\_, [820](#)
  - ddot\_, [819](#)
  - dgemm\_, [822](#)
  - dgemv\_, [819](#)
  - dger\_, [820](#)
  - dnrm2\_, [819](#)
  - dscal\_, [820](#)
  - dtrmm\_, [821](#)
  - dtrsm\_, [821](#)
  - idamax\_, [819](#)
  - saxpy\_, [818](#)
  - scopy\_, [820](#)
  - sdot\_, [819](#)
  - sgemm\_, [822](#)
  - sgemv\_, [819](#)
  - sger\_, [820](#)
  - sscal\_, [821](#)
  - strmm\_, [821](#)
  - strsm\_, [821](#)
- config.h, [801](#), [804](#)
  - \_\_FFLASFFPACK\_HAVE\_BIG\_ENDIAN, [805](#)
  - \_\_FFLASFFPACK\_HAVE\_BLAS, [805](#)
  - \_\_FFLASFFPACK\_HAVE\_CBLAS, [805](#)
  - \_\_FFLASFFPACK\_HAVE\_CXX11, [805](#)
  - \_\_FFLASFFPACK\_HAVE\_DLFCN\_H, [805](#)
  - \_\_FFLASFFPACK\_HAVE\_FLOAT\_H, [805](#)
  - \_\_FFLASFFPACK\_HAVE\_INT128, [805](#)
  - \_\_FFLASFFPACK\_HAVE\_INTPTR\_T\_H, [805](#)
  - \_\_FFLASFFPACK\_HAVE\_LAPACK, [805](#)
  - \_\_FFLASFFPACK\_HAVE\_LIMITS\_H, [805](#)
  - \_\_FFLASFFPACK\_HAVE\_PTHREAD\_H, [805](#)
  - \_\_FFLASFFPACK\_HAVE\_STDDEF\_H, [806](#)
  - \_\_FFLASFFPACK\_HAVE\_STDINT\_H, [806](#)
  - \_\_FFLASFFPACK\_HAVE\_STDIO\_H, [806](#)
  - \_\_FFLASFFPACK\_HAVE\_STDLIB\_H, [806](#)
  - \_\_FFLASFFPACK\_HAVE\_STRINGS\_H, [806](#)
  - \_\_FFLASFFPACK\_HAVE\_STRING\_H, [806](#)
  - \_\_FFLASFFPACK\_HAVE\_SYS\_STAT\_H, [806](#)
  - \_\_FFLASFFPACK\_HAVE\_SYS\_TIME\_H, [806](#)
  - \_\_FFLASFFPACK\_HAVE\_SYS\_TYPES\_H, [806](#)
  - \_\_FFLASFFPACK\_HAVE\_UNISTD\_H, [806](#)
  - \_\_FFLASFFPACK\_LT\_OBJDIR, [806](#)

- \_\_FFLASFFPACK\_OPENBLAS\_NUM\_THREADS, 806
- \_\_FFLASFFPACK\_PACKAGE, 806
- \_\_FFLASFFPACK\_PACKAGE\_BUGREPORT, 806
- \_\_FFLASFFPACK\_PACKAGE\_NAME, 806
- \_\_FFLASFFPACK\_PACKAGE\_STRING, 807
- \_\_FFLASFFPACK\_PACKAGE\_TARNAME, 807
- \_\_FFLASFFPACK\_PACKAGE\_URL, 807
- \_\_FFLASFFPACK\_PACKAGE\_VERSION, 807
- \_\_FFLASFFPACK\_SIZEOF\_CHAR, 807
- \_\_FFLASFFPACK\_SIZEOF\_INT, 807
- \_\_FFLASFFPACK\_SIZEOF\_LONG, 807
- \_\_FFLASFFPACK\_SIZEOF\_LONG\_LONG, 807
- \_\_FFLASFFPACK\_SIZEOF\_SHORT, 807
- \_\_FFLASFFPACK\_SIZEOF\_\_INT64\_T, 807
- \_\_FFLASFFPACK\_STDC\_HEADERS, 807
- \_\_FFLASFFPACK\_USE\_OPENMP, 807
- \_\_FFLASFFPACK\_VERSION, 807
- HAVE\_BIG\_ENDIAN, 801
- HAVE\_BLAS, 801
- HAVE\_CBLAS, 802
- HAVE\_CXX11, 802
- HAVE\_DLFCN\_H, 802
- HAVE\_FLOAT\_H, 802
- HAVE\_INT128, 802
- HAVE\_INTTYPES\_H, 802
- HAVE\_LAPACK, 802
- HAVE\_LIMITS\_H, 802
- HAVE\_PTHREAD\_H, 802
- HAVE\_STDDEF\_H, 802
- HAVE\_STDINT\_H, 802
- HAVE\_STDIO\_H, 802
- HAVE\_STDLIB\_H, 802
- HAVE\_STRING\_H, 802
- HAVE\_STRINGS\_H, 802
- HAVE\_SYS\_STAT\_H, 803
- HAVE\_SYS\_TIME\_H, 803
- HAVE\_SYS\_TYPES\_H, 803
- HAVE\_UNISTD\_H, 803
- LT\_OBJDIR, 803
- OPENBLAS\_NUM\_THREADS, 803
- PACKAGE, 803
- PACKAGE\_BUGREPORT, 803
- PACKAGE\_NAME, 803
- PACKAGE\_STRING, 803
- PACKAGE\_TARNAME, 803
- PACKAGE\_URL, 803
- PACKAGE\_VERSION, 803
- SIZEOF\_\_INT64\_T, 804
- SIZEOF\_CHAR, 803
- SIZEOF\_INT, 803
- SIZEOF\_LONG, 804
- SIZEOF\_LONG\_LONG, 804
- SIZEOF\_SHORT, 804
- STDC\_HEADERS, 804
- USE\_OPENMP, 804
- VERSION, 804
- Configuring and Installing FFLAS-FFPACK, 2
- CONST
  - fflas\_simd.h, 871
- ConstElement\_ptr
  - benchmark-fgemm-rns.C, 785
  - rns\_double, 525
  - rns\_double\_extended, 537
  - RNSInteger< RNS >, 541
  - RNSIntegerMod< RNS >, 544
- CONSTREFERENCE
  - parallel.h, 1032
- convert
  - rns\_double, 527, 528
  - rns\_double\_extended, 538
  - RNSInteger< RNS >, 542
  - RNSIntegerMod< RNS >, 547
- convert\_transpose
  - rns\_double, 527
- ConvertTo< T >, 434
- COO
  - FFLAS, 76
- Coo
  - Coo< Field >, 436
  - Coo< ValT, IdxT >, 434, 435, 437, 438
- coo
  - HelperFlag, 472
- Coo< Field >, 435
  - col, 436
  - Coo, 436
  - deleted, 437
  - operator=, 436
  - row, 437
  - val, 436
- Coo< ValT, IdxT >, 434, 437
  - col, 435, 438
  - Coo, 434, 435, 437, 438
  - operator=, 435, 438
  - row, 435, 438
  - Self, 434, 437
  - val, 435, 438
- coo.h, 886
- coo\_spmv.inl, 887
  - \_\_FFLASFFPACK\_fflas\_sparse\_coo\_spmv\_INL, 888
- coo\_spmv.inl, 888
  - \_\_FFLASFFPACK\_fflas\_sparse\_coo\_spmv\_INL, 889
- coo\_utils.inl, 889
  - \_\_FFLASFFPACK\_fflas\_sparse\_coo\_utils\_INL, 889
- COO\_ZO
  - FFLAS, 76
- CooMat< Field >, 438
  - \_coo16, 439
  - \_coo16\_zo, 439
  - \_coo32, 439
  - \_coo32\_zo, 439
  - \_coo64, 439
  - \_coo64\_zo, 439



- Copying and Licence, [2](#)
- count\_nonconst\_lvalue\_reference< const T &, O... >, [439](#)
  - n, [439](#)
- count\_nonconst\_lvalue\_reference< T >, [439](#)
- count\_nonconst\_lvalue\_reference< T &, O... >, [440](#)
  - n, [440](#)
- count\_nonconst\_lvalue\_reference< T, O... >, [440](#)
  - n, [440](#)
- count\_nonconst\_lvalue\_reference<>, [440](#)
  - n, [440](#)
- CROUT
  - ffpack\_pluq.inl, [940](#)
- CSC
  - FFLAS, [76](#)
- CSC\_ZO
  - FFLAS, [76](#)
- CSR
  - FFLAS, [76](#)
- csr
  - HelperFlag, [472](#)
- csr.h, [889](#)
- CSR\_HYB
  - FFLAS, [76](#)
- csr\_hyb.h, [894](#)
- csr\_hyb\_pspmm.inl, [894](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_pspmm\_INL, [895](#)
- csr\_hyb\_pspmv.inl, [895](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_pspmv\_INL, [895](#)
- csr\_hyb\_spm.inl, [895](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_spm\_INL, [896](#)
- csr\_hyb\_spmv.inl, [896](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_spmv\_INL, [896](#)
- csr\_hyb\_utils.inl, [896](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_HYB\_utils\_INL, [897](#)
- csr\_pspmm.inl, [890](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_pspmm\_INL, [890](#)
- csr\_pspmv.inl, [891](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_pspmv\_INL, [891](#)
- csr\_spm.inl, [891](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_spm\_INL, [892](#)
- csr\_spmv.inl, [892](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_CSR\_spmv\_INL, [893](#)
- csr\_utils.inl, [893](#)
- CSR\_ZO
  - FFLAS, [76](#)
- CsrMat< Field >, [440](#)
  - \_csr16, [441](#)
  - \_csr16\_zo, [441](#)
  - \_csr32, [441](#)
  - \_csr32\_zo, [441](#)
  - \_csr64, [441](#)
  - \_csr64\_zo, [441](#)
- cst
  - Sparse< \_Field, SparseMatrix\_t::COO\_ZO >, [736](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR\_ZO >, [741](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd\_ZO >, [745](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_ZO >, [747](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, [753](#)
- ctz
  - bit\_manipulation.h, [1040](#)
- CUBE
  - arithprog.C, [771](#)
  - benchmark-checkers.C, [777](#)
  - benchmark-fsytrf.C, [793](#)
  - benchmark-ftrtri.C, [795](#)
  - benchmark-inverse.C, [796](#)
  - benchmark-lqup.C, [797](#)
  - benchmark-pluq.C, [798](#)
  - benchmark-wino.C, [800](#)
  - charpoly.C, [808](#)
  - fsyrk.C, [772](#)
  - fsytrf.C, [773](#)
  - ftrtri.C, [774](#)
  - cuda.C, [1049](#)
  - main, [1049](#)
  - Parallel< C, P >, [519](#)
  - cyclic\_shift\_col
    - FFPACK, [367](#), [370](#)
  - cyclic\_shift\_col\_modular\_double
    - ffpack.C, [988](#)
    - ffpack\_c.h, [1009](#)
  - cyclic\_shift\_mathPerm
    - FFPACK, [366](#)
    - ffpack.C, [988](#)
    - ffpack\_c.h, [1009](#)
  - cyclic\_shift\_row
    - FFPACK, [366](#), [369](#)
  - cyclic\_shift\_row\_col
    - FFPACK, [366](#), [369](#)
  - cyclic\_shift\_row\_modular\_double
    - ffpack.C, [988](#)
    - ffpack\_c.h, [1009](#)
- Danilevski
  - FFPACK, [356](#)
  - FFPACK::Protected, [399](#)
- dasum\_
  - config-blas.h, [819](#)
- dat
  - Sparse< \_Field, SparseMatrix\_t::COO >, [734](#)
  - Sparse< \_Field, SparseMatrix\_t::COO\_ZO >, [736](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR >, [738](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR\_HYB >, [739](#)



- Sparse< \_Field, SparseMatrix\_t::CSR\_ZO >, 742
- Sparse< \_Field, SparseMatrix\_t::ELL >, 743
- Sparse< \_Field, SparseMatrix\_t::ELL\_simd >, 745
- Sparse< \_Field, SparseMatrix\_t::ELL\_simd\_ZO >, 746
- Sparse< \_Field, SparseMatrix\_t::ELL\_ZO >, 748
- Sparse< \_Field, SparseMatrix\_t::HYB\_ZO >, 749
- Sparse< \_Field, SparseMatrix\_t::SELL >, 752
- Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, 754
- data
  - Argument, 408
- daxpy\_
  - config-blas.h, 818
- dcopy\_
  - config-blas.h, 820
- ddot\_
  - config-blas.h, 819
- debug.h, 1041
  - FFLASFFPACK\_abort, 1042
  - FFLASFFPACK\_check, 1041
- DeCompressRows
  - FFPACK::Protected, 402
- DeCompressRowsQA
  - FFPACK::Protected, 402
- DeCompressRowsQK
  - FFPACK::Protected, 402
- DefaultBoundedTag, 441
- DefaultTag, 441
- delayed
  - RNSIntegerMod< RNS >, 545
  - Sparse< \_Field, SparseMatrix\_t::COO >, 734
  - Sparse< \_Field, SparseMatrix\_t::COO\_ZO >, 736
  - Sparse< \_Field, SparseMatrix\_t::CSR >, 737
  - Sparse< \_Field, SparseMatrix\_t::CSR\_HYB >, 739
  - Sparse< \_Field, SparseMatrix\_t::CSR\_ZO >, 741
  - Sparse< \_Field, SparseMatrix\_t::ELL >, 742
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd >, 744
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd\_ZO >, 745
  - Sparse< \_Field, SparseMatrix\_t::ELL\_ZO >, 747
  - Sparse< \_Field, SparseMatrix\_t::HYB\_ZO >, 749
  - Sparse< \_Field, SparseMatrix\_t::SELL >, 750
  - Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, 753
- DelayedField
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, 497
- delayedField
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, 500
- DelayedField\_t
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, 497
- DelayedTag, 442
- deleted
  - Coo< Field >, 437
- DENSE\_THRESHOLD
  - fflas\_sparse.h, 884
- denseCols
  - StatsMatrix, 757
- denseRows
  - StatsMatrix, 757
- Det
  - FFPACK, 332, 333, 356, 357, 376, 377
- det.C, 809
  - main, 809
- Det\_modular\_double
  - ffpack.C, 998
  - ffpack\_c.h, 1017
- deviationCol
  - StatsMatrix, 756
- deviationColDifference
  - StatsMatrix, 756
- deviationRow
  - StatsMatrix, 756
- deviationRowDifference
  - StatsMatrix, 756
- DFElt
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, 497
- dgemm\_
  - config-blas.h, 822
  - fblas.C, 1050
- dgemv\_
  - config-blas.h, 819
- dger\_
  - config-blas.h, 820
- digits
  - limits< char >, 486
  - limits< double >, 487
  - limits< float >, 487
  - limits< int >, 488
  - limits< long >, 489
  - limits< long long >, 490
  - limits< short int >, 491
  - limits< signed char >, 492
  - limits< unsigned char >, 493
  - limits< unsigned int >, 493
  - limits< unsigned long >, 494
  - limits< unsigned long long >, 494
  - limits< unsigned short int >, 495
- div
  - ScalFunctions< Element >, 554
  - Simd256\_impl< true, false, true, 8 >, 626
  - Simd512\_impl< true, false, true, 8 >, 708
- DivideAndConquer, 442
- dnrm2\_
  - config-blas.h, 819
- DNS\_BIN\_VER
  - read\_sparse.h, 907
- doApplyS
  - FFPACK, 362
- doApplyT
  - FFPACK, 363
- doBenchs
  - Bench< Elt >, 412

doTests  
     Test< Elt >, 761  
 DotProdBoundClassic  
     FFLAS::Protected, 216  
 DOUBLE\_TO\_FLOAT\_CROSSOVER  
     fflas.h, 826  
     winograd.C, 775  
 dscal\_  
     config-blas.h, 820  
 dtrmm\_  
     config-blas.h, 821  
 dtrsm\_  
     config-blas.h, 821  
 DynamicPeeling  
     FFLAS::Protected, 221  
 DynamicPeeling2  
     FFLAS::Protected, 221  
 EFGFF  
     benchmark-dsytrf.C, 780  
 Element  
     FieldSimd< \_Field >, 451  
     readMyMachineType< Field, mpz\_t >, 523  
     readMyMachineType< Field, T >, 523  
     rns\_double, 525  
     rns\_double\_extended, 537  
     RNSInteger< RNS >, 540  
     RNSIntegerMod< RNS >, 544  
     TestOneMethod< Simd >, 763  
 Element\_ptr  
     benchmark-fgemm-rns.C, 785  
     readMyMachineType< Field, mpz\_t >, 523  
     readMyMachineType< Field, T >, 523  
     rns\_double, 525  
     rns\_double\_extended, 537  
     RNSInteger< RNS >, 541  
     RNSIntegerMod< RNS >, 544  
 ElementTraits< double >, 442  
     value, 443  
 ElementTraits< Element >, 442  
     value, 442  
 ElementTraits< FFPACK::rns\_double\_elt >, 443  
     value, 443  
 ElementTraits< float >, 443  
     value, 443  
 ElementTraits< Givaro::Integer >, 443  
     value, 443  
 ElementTraits< int16\_t >, 444  
     value, 444  
 ElementTraits< int32\_t >, 444  
     value, 444  
 ElementTraits< int64\_t >, 444  
     value, 444  
 ElementTraits< int8\_t >, 444  
     value, 445  
 ElementTraits< Reclnt::rint< K > >, 445  
     value, 445  
 ElementTraits< Reclnt::rmin< K, MG > >, 445  
     value, 445  
 ElementTraits< Reclnt::ruint< K > >, 445  
     value, 446  
 ElementTraits< uint16\_t >, 446  
     value, 446  
 ElementTraits< uint32\_t >, 446  
     value, 446  
 ElementTraits< uint64\_t >, 446  
     value, 446  
 ElementTraits< uint8\_t >, 447  
     value, 447  
 ELL  
     FFLAS, 76  
 ell  
     HelperFlag, 472  
 ell.h, 897  
 ell\_pspmm.inl, 897  
     \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_pspmm\_INL,  
         898  
 ell\_pspmv.inl, 898  
     \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_pspmv\_INL,  
         899  
 ELL\_simd  
     FFLAS, 76  
 ell\_simd.h, 901  
 ell\_simd\_pspmv.inl, 902  
     \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_simd\_pspmv\_INL,  
         902  
 ell\_simd\_spmv.inl, 902  
     \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_simd\_spmv\_INL,  
         903  
 ell\_simd\_utils.inl, 903  
     \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_simd\_utils\_INL,  
         904  
 ELL\_simd\_ZO  
     FFLAS, 76  
 ell\_spm.inl, 899  
     \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_spm\_INL,  
         900  
 ell\_spmv.inl, 900  
     \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_spmv\_INL,  
         901  
 ell\_utils.inl, 901  
     \_\_FFLASFFPACK\_fflas\_sparse\_ELL\_utils\_INL,  
         901  
 ELL\_ZO  
     FFLAS, 76  
 EllMat< Field >, 447  
     \_ell16, 447  
     \_ell16\_zo, 447  
     \_ell32, 447  
     \_ell32\_zo, 447  
     \_ell64, 447  
     \_ell64\_zo, 447  
 Ell\_ptr  
     Bench< Elt >, 411  
     Test< Elt >, 760  
 ENABLE\_ALL\_CHECKINGS  
     benchmark-checkers.C, 777

- ffpack\_ftrtr.inl, [934](#)
- test-fdot.C, [1059](#)
- test-fgemm-check.C, [1061](#)
- test-fsyr2k.C, [1071](#)
- test-fsyrk.C, [1073](#)
- test-ftrmv.C, [1077](#)
- test-ftrsm-check.C, [1078](#)
- test-ftrsm.C, [1079](#)
- test-ftrssyr2k.C, [1080](#)
- test-ftrstr.C, [1081](#)
- test-ftrsv.C, [1082](#)
- test-ftrtri.C, [1083](#)
- test-invert-check.C, [1085](#)
- test-pluq-check.C, [1094](#)
- ENABLE\_CHECKER\_charpoly
  - test-charpoly-check.C, [1052](#)
- ENABLE\_CHECKER\_Det
  - test-det-check.C, [1054](#)
- ENABLE\_CHECKER\_fgemm
  - test-fgemm.C, [1062](#)
- enable\_if\_no\_simd\_t
  - Bench< Elt >, [411](#)
  - Test< Elt >, [760](#)
- enable\_if\_simd128\_t
  - Bench< Elt >, [411](#)
  - Test< Elt >, [760](#)
- enable\_if\_simd256\_t
  - Bench< Elt >, [411](#)
  - Test< Elt >, [760](#)
- enable\_if\_simd512\_t
  - Bench< Elt >, [411](#)
  - Test< Elt >, [761](#)
- enable\_if\_t
  - Bench< Elt >, [411](#)
  - Test< Elt >, [760](#)
  - TestOneMethod< Simd >, [763](#)
- END\_OF\_ARGUMENTS
  - args-parser.h, [1039](#)
- END\_PARALLEL\_MAIN
  - parallel.h, [1032](#)
- eq
  - ScalFunctions< Element >, [555](#)
  - Simd128\_impl< true, true, false, 2 >, [571](#)
  - Simd128\_impl< true, true, false, 4 >, [581](#)
  - Simd128\_impl< true, true, false, 8 >, [591](#)
  - Simd128\_impl< true, true, true, 2 >, [600](#)
  - Simd128\_impl< true, true, true, 4 >, [609](#)
  - Simd128\_impl< true, true, true, 8 >, [618](#)
  - Simd256\_impl< true, false, true, 8 >, [627](#)
  - Simd256\_impl< true, true, false, 2 >, [638](#)
  - Simd256\_impl< true, true, false, 4 >, [656](#)
  - Simd256\_impl< true, true, false, 8 >, [666](#)
  - Simd256\_impl< true, true, true, 2 >, [675](#)
  - Simd256\_impl< true, true, true, 4 >, [686](#), [692](#)
  - Simd256\_impl< true, true, true, 8 >, [701](#)
  - Simd512\_impl< true, false, true, 8 >, [709](#)
  - Simd512\_impl< true, true, false, 8 >, [720](#)
  - Simd512\_impl< true, true, true, 8 >, [729](#)
- eval\_func\_on\_array
  - test-simd.C, [1100](#)
- evaluate\_scalar\_method
  - TestOneMethod< Simd >, [763](#), [764](#)
- evaluate\_simd\_method
  - TestOneMethod< Simd >, [764](#)
- example
  - Argument, [407](#)
- ExpandBlockBiDiagonalToBruhat
  - FFPACK, [350](#)
- expandLCRE
  - FFPACK, [354](#)
- F
  - Bench< Elt >, [412](#)
  - Test< Elt >, [762](#)
- fadd
  - FFLAS, [78](#), [80](#), [82](#), [166](#), [173](#), [174](#)
  - FFLAS::details, [203](#), [204](#)
- fadd\_1\_modular\_double
  - fflas\_c.h, [955](#)
  - fflas\_lvl1.C, [974](#)
- fadd\_2\_modular\_double
  - fflas\_c.h, [958](#)
  - fflas\_lvl2.C, [979](#)
- faddin
  - FFLAS, [78](#), [81](#), [166](#), [175](#)
- faddin\_1\_modular\_double
  - fflas\_c.h, [955](#)
  - fflas\_lvl1.C, [975](#)
- faddin\_2\_modular\_double
  - fflas\_c.h, [959](#)
  - fflas\_lvl2.C, [979](#)
- Failure, [448](#)
  - \_errorStream, [449](#)
  - Failure, [448](#)
  - operator(), [448](#)
  - print, [449](#)
  - setErrorStream, [449](#)
- failure
  - FFPACK, [382](#)
- FailureCharpolyCheck, [449](#)
- FailureDetCheck, [449](#)
- FailureFgemmCheck, [449](#)
- FailureInvertCheck, [450](#)
- FailurePLUQCheck, [450](#)
- FailureTrsmCheck, [450](#)
- fassign
  - FFLAS, [82–84](#), [162](#), [166](#)
- fassign\_1\_modular\_double
  - fflas\_c.h, [954](#)
  - fflas\_lvl1.C, [973](#)
- fassign\_2\_modular\_double
  - fflas\_c.h, [956](#)
  - fflas\_lvl2.C, [976](#)
- faxpby
  - FFLAS, [135](#), [141](#)
- faxpy
  - FFLAS, [86](#), [87](#), [164](#), [172](#)

- FFLAS::details, 205
- faxpy\_1\_modular\_double
  - fflas\_c.h, 954
  - fflas\_lvl1.C, 974
- faxpy\_2\_modular\_double
  - fflas\_c.h, 958
  - fflas\_lvl2.C, 978
- fblas.C, 1049
  - \_\_FFLASFFPACK\_CONFIGURATION, 1050
  - dgemm\_, 1050
  - main, 1050
- fconvert
  - FFLAS, 132, 140, 160
- fconvert\_rns
  - FFLAS, 157, 158
- fconvert\_trans\_rns
  - FFLAS, 158
- fdot
  - FFLAS, 87–89, 135, 164, 184
- fdot\_1\_modular\_double
  - fflas\_c.h, 954
  - fflas\_lvl1.C, 974
- fequal
  - FFLAS, 134, 138, 162, 167
- fequal\_1\_modular\_double
  - fflas\_c.h, 953
  - fflas\_lvl1.C, 973
- fequal\_2\_modular\_double
  - fflas\_c.h, 956
  - fflas\_lvl2.C, 977
- FFLAS, 42, 45
  - alignable, 191
  - alignable< Givaro::Integer \* >, 192
  - BaseTimer, 74
  - bitsize, 142
  - bitsize< Givaro::ZRing< Givaro::Integer > >, 142
  - BlockCuts, 181, 183
  - BlockCuts< CuttingStrategy::Block, StrategyParameter::Fixed >, 183
  - BlockCuts< CuttingStrategy::Block, StrategyParameter::Grain >, 182
  - BlockCuts< CuttingStrategy::Block, StrategyParameter::Threads >, 183
  - BlockCuts< CuttingStrategy::Column, StrategyParameter::Fixed >, 182
  - BlockCuts< CuttingStrategy::Column, StrategyParameter::Grain >, 182
  - BlockCuts< CuttingStrategy::Column, StrategyParameter::Threads >, 183
  - BlockCuts< CuttingStrategy::Row, StrategyParameter::Fixed >, 182
  - BlockCuts< CuttingStrategy::Row, StrategyParameter::Grain >, 182
  - BlockCuts< CuttingStrategy::Row, StrategyParameter::Threads >, 183
  - BlockCuts< CuttingStrategy::Single, StrategyParameter::Threads >, 181
  - cblas\_imptrsm, 130
  - Checker\_fgemm, 72
  - Checker\_ftrsm, 72
  - computeDeviation, 155
  - computeS1S2, 125
  - COO, 76
  - COO\_ZO, 76
  - CSC, 76
  - CSC\_ZO, 76
  - CSR, 76
  - CSR\_HYB, 76
  - CSR\_ZO, 76
  - ELL, 76
  - ELL\_simd, 76
  - ELL\_simd\_ZO, 76
  - ELL\_ZO, 76
  - fadd, 78, 80, 82, 166, 173, 174
  - faddin, 78, 81, 166, 175
  - fassign, 82–84, 162, 166
  - faxpby, 135, 141
  - faxpy, 86, 87, 164, 172
  - fconvert, 132, 140, 160
  - fconvert\_rns, 157, 158
  - fconvert\_trans\_rns, 158
  - fdot, 87–89, 135, 164, 184
  - fequal, 134, 138, 162, 167
  - FFLAS\_BASE, 76
  - fflas\_delete, 156, 192
  - FFLAS\_DIAG, 75
  - FFLAS\_FORMAT, 76
  - fflas\_new, 157, 158, 192
  - FFLAS\_ORDER, 74
  - FFLAS\_SIDE, 75
  - FFLAS\_TRANSPOSE, 74
  - FFLAS\_UPLO, 75
  - FflasAuto, 77
  - FflasBinary, 77
  - FflasColMajor, 74
  - FflasDense, 77
  - FflasDouble, 76
  - FflasFloat, 76
  - FflasGeneric, 76
  - FflasLeft, 75
  - FflasLeftTri, 75
  - FflasLower, 75
  - FflasMaple, 77
  - FflasMath, 77
  - FflasNonUnit, 75
  - FflasNoTrans, 75
  - FflasRight, 75
  - FflasRightTri, 75
  - FflasRowMajor, 74
  - FflasSageMath, 77
  - FflasSMS, 77
  - FflasTrans, 75
  - FflasUnit, 75
  - FflasUpper, 75
  - fgemm, 89–91, 93–98, 146, 179, 180
  - fgemv, 98–103, 175, 187

- fgcr, 104–107, 176
- fidentity, 139, 168
- finit, 109, 112, 131, 139, 160, 170
- finit\_rns, 157, 158
- finit\_trans\_rns, 157
- fiszero, 134, 138, 162, 168
- fmove, 142, 172
- fneg, 132, 141, 161, 170
- fnegin, 132, 140, 160, 170
- ForceCheck\_fgemm, 72
- ForceCheck\_ftsm, 72
- frand, 133, 138
- freduce, 108–110, 113, 159, 169
- freduce\_constoverride, 109, 112
- freivalds, 113
- fscal, 114–118, 163, 171
- fscalin, 114–118, 163, 171
- fspmm, 147
- fspmv, 147, 154
- fsquare, 92, 93, 181
- fsub, 78, 81, 166, 174
- fsubin, 78, 82, 174
- fswap, 135, 165
- fsyr2k, 118
- fsyrk, 119–126
- fsyrk\_strassen, 126, 144
- ftmm, 127, 128, 178
- ftmv, 143
- ftsm, 128–130, 143, 147, 177
- ftsv, 130, 177
- fzero, 133, 137, 161, 167
- getArgumentValue, 188
- getDataType, 153, 154
- getSeed, 193
- getStat, 156
- getTLBSize, 193
- has\_equal, 73
- has\_minus, 73
- has\_minus\_eq, 73
- has\_mul, 73
- has\_mul\_eq, 74
- has\_plus, 73
- has\_plus\_eq, 73
- HYB\_ZO, 76
- igemm\_, 131
- InfNorm, 77
- max3, 77
- max4, 77
- maxCardinality, 156
- maxCardinality< Givaro::Modular< int32\_t > >, 156
- maxCardinality< Givaro::Modular< int64\_t > >, 156
- min3, 77
- min4, 77
- minCardinality, 156
- MKLSparseMatrixFormat, 73
- mone, 76
- NoSimdSparseMatrix, 72
- NotMKLSparseMatrixFormat, 73
- NotZOSparseMatrix, 72
- number\_kind, 76
- one, 76
- operator<<, 153
- other, 76
- parseArguments, 188
- pfadd, 79
- pfaddin, 79
- pfgemm, 144, 184–186
- pfgemm\_1D\_rec, 145
- pfgemm\_2D\_rec, 145
- pfgemm\_3D\_rec, 145
- pfgemm\_3D\_rec2, 146
- pfrand, 184
- pfreduce, 110
- pfsb, 79
- pfsbin, 80
- pfzero, 184
- preamble, 190
- prefetch, 192
- queryCacheSizes, 193
- queryL1CacheSize, 193
- queryTopLevelCacheSize, 193
- readDnsFormat, 154
- readMachineType, 154
- ReadMatrix, 190
- readSmsFormat, 153
- readSprFormat, 153
- SELL, 76
- SELL\_ZO, 76
- SimdSparseMatrix, 72
- sparse\_delete, 148–152, 155
- sparse\_init, 148–153, 155
- sparse\_print, 149, 152, 155
- SparseMatrix\_t, 76
- SysTimer, 74
- Timer, 74
- UserTimer, 74
- writeCommandString, 188
- writeDnsFormat, 154
- WriteMatrix, 188, 191
- WritePermutation, 191
- zero, 76
- ZOSparseMatrix, 72
- fflas-101\_1.C, 1104
  - main, 1104
- fflas-101\_3.C, 1104
  - main, 1104
- FFLAS-FFPACK, 41
- FFLAS-FFPACK Documentation., 1
- FFLAS-FFPACK fields, 43
- fflas-ffpack-config.h, 823
  - GCC\_VERSION, 823
- fflas-ffpack-default-thresholds.h, 823
  - \_\_FFLASFFPACK\_ARITHPROG\_THRESHOLD, 824

- \_\_FFLASFFPACK\_CHARPOLY\_Danilevskii\_LUKrylov\_THRESHOLD, 824
- \_\_FFLASFFPACK\_CHARPOLY\_LUKrylov\_ArithProg\_THRESHOLD, 824
- \_\_FFLASFFPACK\_FSYRK\_THRESHOLD, 824
- \_\_FFLASFFPACK\_FSYTRF\_THRESHOLD, 824
- \_\_FFLASFFPACK\_FTRTRI\_THRESHOLD, 824
- \_\_FFLASFFPACK\_PLUQ\_THRESHOLD, 824
- \_\_FFLASFFPACK\_WINOTHRESHOLD, 823
- \_\_FFLASFFPACK\_WINOTHRESHOLD\_BAL, 823
- \_\_FFLASFFPACK\_WINOTHRESHOLD\_BAL\_FLT, 824
- \_\_FFLASFFPACK\_WINOTHRESHOLD\_FLT, 823
- fflas-ffpack-thresholds.h, 824
- fflas-ffpack.doxy, 824
- fflas-ffpack.h, 824
- fflas.doxy, 824
- fflas.h, 824
  - DOUBLE\_TO\_FLOAT\_CROSSOVER, 826
  - WINOTHRESHOLD, 826
- FFLAS::\_ftranspose\_impl, 194
  - nonsquare\_inplace\_v1, 194
  - nonsquare\_inplace\_v2, 194
  - not\_inplace, 194
  - square\_inplace, 194
- FFLAS::BLAS3, 195
  - Bini, 196
  - Winograd, 197
  - Winograd\_L\_S, 200
  - Winograd\_LR\_S, 200
  - Winograd\_R\_S, 201
  - WinogradAcc\_2\_24, 198
  - WinogradAcc\_2\_27, 198
  - WinogradAcc\_3\_21, 198
  - WinogradAcc\_3\_23, 197
  - WinogradAcc\_L\_S, 200
  - WinogradAcc\_LR, 199
  - WinogradAcc\_R\_S, 199
  - WinoPar, 197
- FFLAS::csr\_hyb\_details, 201
- FFLAS::CuttingStrategy, 201
  - RNSModulus, 202
- FFLAS::details, 202
  - BlockingFactor, 210
  - fadd, 203, 204
  - faxpy, 205
  - freduce, 205, 206
  - fscal, 206, 207
  - fscaln, 206, 207
  - gebp, 210
  - igebb11, 209
  - igebb14, 208
  - igebb21, 208
  - igebb24, 207
  - igebb41, 208
  - igebb44, 207
  - igebp, 209
  - pack\_lhs, 209
  - pack\_rhs, 209
  - FFLAS::details\_spmv, 210
  - FFLAS::FieldCategories, 211
  - FFLAS::FieldCategories, 211
  - FFLAS::MMHelperAlgo, 211
  - FFLAS::ModeCategories, 212
  - FFLAS::ParSeqHelper, 212
  - FFLAS::Protected, 213
    - computeFactorClassic, 216
    - DotProdBoundClassic, 216
    - DynamicPeeling, 221
    - DynamicPeeling2, 221
    - fgemm\_convert, 217
    - fgemv\_convert, 222
    - fger\_convert, 223
    - fsquareCommon, 220
    - fsyrk\_convert, 223
    - igemm, 226
    - igemm\_colmajor, 226
    - MatF2MatD\_Triangular, 227
    - MatF2MatFI\_Triangular, 227
    - min\_types, 224, 225
    - NeedDoublePreAddReduction, 219
    - NeedPreAddReduction, 218
    - NeedPreAxpPyReduction, 224
    - NeedPreScalReduction, 223, 224
    - NeedPreSubReduction, 218
    - ScalAndReduce, 219, 223
    - TRSMBound, 216, 217
    - unfit, 225, 226
    - WinogradCalc, 222
    - WinogradSteps, 221
    - WinogradThreshold, 220
  - FFLAS::sell\_details, 227
  - FFLAS::sparse\_details, 228
    - fspmm, 234–236
    - fspmm\_dispatch, 233, 234
    - fspmv, 231–233, 241
    - fspmv\_dispatch, 231
    - init\_y, 231
    - pfspmm, 237–239
    - pfspmm\_dispatch, 236, 237
    - pfspmv, 239, 240
  - FFLAS::sparse\_details\_impl, 242
    - fspmm, 250, 257, 258, 263, 264, 268, 277
    - fspmm\_mone, 251, 259, 269
    - fspmm\_mone\_simd\_aligned, 252, 259, 270
    - fspmm\_mone\_simd\_unaligned, 252, 260, 270
    - fspmm\_one, 251, 258, 269
    - fspmm\_one\_simd\_aligned, 252, 259, 270
    - fspmm\_one\_simd\_unaligned, 252, 259, 270
    - fspmm\_simd\_aligned, 251, 258
    - fspmm\_simd\_unaligned, 251, 258
    - fspmv, 253, 260, 264, 271, 273, 274, 278, 280
    - fspmv\_mone, 253, 254, 261, 271, 272, 275, 281, 282
    - fspmv\_mone\_simd, 275, 281

- fspmv\_one, [253](#), [254](#), [260](#), [261](#), [271](#), [272](#), [274](#), [275](#), [281](#)
  - fspmv\_one\_simd, [275](#), [281](#)
  - fspmv\_simd, [274](#), [280](#)
  - pfspmm, [254](#), [261](#), [262](#), [264–266](#), [276](#)
  - pfspmm\_mone, [255](#)
  - pfspmm\_one, [255](#)
  - pfspmm\_zo, [266](#)
  - pfspmv, [256](#), [263](#), [266](#), [267](#), [272](#), [276–279](#)
  - pfspmv\_mone, [257](#), [267](#), [268](#), [273](#), [279](#)
  - pfspmv\_one, [256](#), [257](#), [267](#), [273](#), [279](#)
  - pfspmv\_task, [256](#)
- FFLAS::StrategyParameter, [282](#)
- FFLAS::StructureHelper, [282](#)
- FFLAS::vectorised, [283](#)
  - add, [285](#)
  - addp, [284](#)
  - axpyp, [285](#)
  - modp, [287](#)
  - reduce, [286](#), [287](#)
  - scalp, [287](#), [288](#)
  - sub, [285](#)
  - subp, [285](#)
  - VEC\_ADD, [284](#)
  - VEC\_SUB, [284](#)
- FFLAS::vectorised::unswitch, [288](#)
  - axpyp, [289](#)
  - modp, [289](#), [290](#)
  - scalp, [290](#)
- fflas\_101.C, [1104](#)
  - main, [1105](#)
- fflas\_101\_lvl1.C, [1105](#)
  - main, [1105](#)
- FFLAS\_BASE
  - FFLAS, [76](#)
- fflas\_bounds.inl, [826](#)
  - \_\_FFLASFFPACK\_fflas\_bounds\_INL, [827](#)
  - FFLAS\_INT\_TYPE, [827](#)
- fflas\_c.h, [949](#)
  - fadd\_1\_modular\_double, [955](#)
  - fadd\_2\_modular\_double, [958](#)
  - faddin\_1\_modular\_double, [955](#)
  - faddin\_2\_modular\_double, [959](#)
  - fassign\_1\_modular\_double, [954](#)
  - fassign\_2\_modular\_double, [956](#)
  - faxpy\_1\_modular\_double, [954](#)
  - faxpy\_2\_modular\_double, [958](#)
  - fdot\_1\_modular\_double, [954](#)
  - fequal\_1\_modular\_double, [953](#)
  - fequal\_2\_modular\_double, [956](#)
  - FFLAS\_C\_BASE, [952](#)
  - FFLAS\_C\_DIAG, [952](#)
  - FFLAS\_C\_ORDER, [951](#)
  - FFLAS\_C\_SIDE, [952](#)
  - FFLAS\_C\_TRANSPOSE, [951](#)
  - FFLAS\_C\_UPLO, [952](#)
  - FFLAS\_COMPILED, [951](#)
  - FflasColMajor, [951](#)
  - FflasDouble, [952](#)
  - FflasFloat, [952](#)
  - FflasGeneric, [952](#)
  - FflasLeft, [952](#)
  - FflasLower, [952](#)
  - FflasNonUnit, [952](#)
  - FflasNoTrans, [952](#)
  - FflasRight, [952](#)
  - FflasRowMajor, [951](#)
  - FflasTrans, [952](#)
  - FflasUnit, [952](#)
  - FflasUpper, [952](#)
  - fgemm\_3\_modular\_double, [960](#)
  - fgemv\_2\_modular\_double, [959](#)
  - fger\_2\_modular\_double, [959](#)
  - fidentity\_2\_modular\_double, [956](#)
  - fiszero\_1\_modular\_double, [953](#)
  - fiszero\_2\_modular\_double, [956](#)
  - fmove\_2\_modular\_double, [958](#)
  - fneg\_1\_modular\_double, [953](#)
  - fneg\_2\_modular\_double, [957](#)
  - fnegin\_1\_modular\_double, [953](#)
  - fnegin\_2\_modular\_double, [957](#)
  - freduce\_1\_modular\_double, [953](#)
  - freduce\_2\_modular\_double, [957](#)
  - freducein\_1\_modular\_double, [953](#)
  - freducein\_2\_modular\_double, [957](#)
  - fscale\_1\_modular\_double, [954](#)
  - fscale\_2\_modular\_double, [957](#)
  - fscalein\_1\_modular\_double, [954](#)
  - fscalein\_2\_modular\_double, [957](#)
  - fsquare\_3\_modular\_double, [961](#)
  - fsub\_1\_modular\_double, [955](#)
  - fsub\_2\_modular\_double, [958](#)
  - fsubin\_1\_modular\_double, [955](#)
  - fsubin\_2\_modular\_double, [959](#)
  - fswap\_1\_modular\_double, [955](#)
  - ftmm\_3\_modular\_double, [960](#)
  - ftsm\_3\_modular\_double, [960](#)
  - ftsv\_2\_modular\_double, [960](#)
  - fzero\_1\_modular\_double, [953](#)
  - fzero\_2\_modular\_double, [956](#)
- FFLAS\_C\_BASE
  - fflas\_c.h, [952](#)
- FFLAS\_C\_DIAG
  - fflas\_c.h, [952](#)
  - ffpack\_c.h, [1007](#)
- FFLAS\_C\_ORDER
  - fflas\_c.h, [951](#)
  - ffpack\_c.h, [1006](#)
- FFLAS\_C\_SIDE
  - fflas\_c.h, [952](#)
  - ffpack\_c.h, [1007](#)
- FFLAS\_C\_TRANSPOSE
  - fflas\_c.h, [951](#)
  - ffpack\_c.h, [1006](#)
- FFLAS\_C\_UPLO
  - fflas\_c.h, [952](#)



- ffpack\_c.h, 1006
- FFLAS\_COMPILED
  - fflas\_c.h, 951
  - ffpack\_inst.C, 1023
  - ffpack\_inst.h, 1024
- fflas\_const\_cast
  - FFPACK, 369, 382
- fflas\_delete
  - FFLAS, 156, 192
- FFLAS\_DIAG
  - FFLAS, 75
- FFLAS\_ELT
  - fflas\_L1\_inst.C, 962
  - fflas\_L1\_inst.h, 963
  - fflas\_L2\_inst.C, 965
  - fflas\_L2\_inst.h, 966
  - fflas\_L3\_inst.C, 969
  - fflas\_L3\_inst.h, 970
  - ffpack\_inst.C, 1023, 1024
  - ffpack\_inst.h, 1024, 1025
- fflas\_enum.h, 827
- fflas\_fadd.h, 827
- fflas\_fadd.inl, 829
  - \_\_FFLASFFPACK\_fadd\_INL, 830
- fflas\_fassign.h, 830
- fflas\_fassign.inl, 830
  - \_\_FFLASFFPACK\_fassign\_INL, 831
- fflas\_faxpy.inl, 831
  - \_\_FFLASFFPACK\_faxpy\_INL, 832
- fflas\_fdot.inl, 832
  - \_\_FFLASFFPACK\_fdot\_INL, 833
- fflas\_fgemm.inl, 833
  - \_\_FFLASFFPACK\_fgemm\_INL, 835
- fflas\_fgemv.inl, 841
  - \_\_FFLASFFPACK\_fgemv\_INL, 843
- fflas\_fgemv\_mp.inl, 843
  - \_\_FFLASFFPACK\_fgemv\_mp\_INL, 844
- fflas\_fger.inl, 844
  - \_\_FFLASFFPACK\_fger\_INL, 845
- fflas\_fger\_mp.inl, 845
  - \_\_FFPACK\_fger\_mp\_INL, 845
- FFLAS\_FIELD
  - fflas\_L1\_inst.C, 961, 962
  - fflas\_L1\_inst.h, 962, 963
  - fflas\_L2\_inst.C, 965
  - fflas\_L2\_inst.h, 966
  - fflas\_L3\_inst.C, 969
  - fflas\_L3\_inst.h, 970
  - ffpack\_inst.C, 1023
  - ffpack\_inst.h, 1024
- FFLAS\_FORMAT
  - FFLAS, 76
- fflas\_freduces.h, 846
- fflas\_freduces.inl, 847
  - \_\_FFLASFFPACK\_fflas\_freduces\_INL, 848
  - FFLASFFPACK\_COPY\_REDUCE, 848
- fflas\_freduces\_mp.inl, 848
  - \_\_FFLASFFPACK\_fflas\_freduces\_mp\_INL, 848
- fflas\_freivalds.inl, 849
  - \_\_FFLASFFPACK\_freivalds\_INL, 849
- fflas\_fscal.h, 849
- fflas\_fscal.inl, 849
  - \_\_FFLASFFPACK\_fscal\_INL, 850
- fflas\_fscal\_mp.inl, 851
  - \_\_FFLASFFPACK\_fscal\_mp\_INL, 851
- fflas\_fsyr2k.inl, 851
  - \_\_FFLASFFPACK\_fflas\_fsyr2k\_INL, 852
- fflas\_fsyrk.inl, 852
  - \_\_FFLASFFPACK\_fflas\_fsyrk\_INL, 854
- fflas\_fsyrk\_strassen.inl, 854
  - \_\_FFLASFFPACK\_fflas\_fsyrk\_strassen\_INL, 855
- fflas\_ftrmm.inl, 855
  - \_\_FFLASFFPACK\_ftrmm\_INL, 855
- fflas\_ftrsm.inl, 855
  - \_\_FFLASFFPACK\_ftrsm\_INL, 856
- fflas\_ftrsm\_mp.inl, 856
  - \_\_FFPACK\_ftrsm\_mp\_INL, 856
- fflas\_ftrsv.inl, 857
  - \_\_FFLASFFPACK\_ftrsv\_INL, 857
- fflas\_helpers.inl, 857
  - \_\_FFLASFFPACK\_fflas\_fflas\_mmhelper\_INL, 858
- FFLAS\_INT\_TYPE
  - fflas\_bounds.inl, 827
- fflas\_intrinsic.h, 1042
- fflas\_io.h, 1042
- fflas\_L1\_inst.C, 961
  - \_\_FFLAS\_L1\_INST\_C, 961
  - FFLAS\_ELT, 962
  - FFLAS\_FIELD, 961, 962
  - INST\_OR\_DECL, 961
- fflas\_L1\_inst.h, 962
  - FFLAS\_ELT, 963
  - FFLAS\_FIELD, 962, 963
  - INST\_OR\_DECL, 962
- fflas\_L1\_inst\_implem.inl, 963
- fflas\_L2\_inst.C, 964
  - \_\_FFLAS\_L2\_INST\_C, 965
  - FFLAS\_ELT, 965
  - FFLAS\_FIELD, 965
  - INST\_OR\_DECL, 965
- fflas\_L2\_inst.h, 965
  - FFLAS\_ELT, 966
  - FFLAS\_FIELD, 966
  - INST\_OR\_DECL, 966
- fflas\_L2\_inst\_implem.inl, 966
- fflas\_L3\_inst.C, 968
  - \_\_FFLAS\_L3\_INST\_C, 968
  - FFLAS\_ELT, 969
  - FFLAS\_FIELD, 969
  - INST\_OR\_DECL, 968
- fflas\_L3\_inst.h, 969
  - FFLAS\_ELT, 970
  - FFLAS\_FIELD, 970
  - INST\_OR\_DECL, 970
- fflas\_L3\_inst\_implem.inl, 970
  - \_\_FFLAS\_\_TRSM\_READONLY, 971



- fflas\_level1.inl, [861](#)
  - \_\_FFLASFFPACK\_fflas\_fflas\_level1\_INL, [864](#)
- fflas\_level2.inl, [864](#)
  - \_\_FFLASFFPACK\_fflas\_fflas\_level2\_INL, [866](#)
- fflas\_level3.inl, [866](#)
  - \_\_FFLASFFPACK\_fflas\_fflas\_level3\_INL, [869](#)
  - \_\_FFLAS\_\_TRSM\_READONLY, [869](#)
- fflas\_lvl1.C, [971](#)
  - fadd\_1\_modular\_double, [974](#)
  - faddin\_1\_modular\_double, [975](#)
  - fassign\_1\_modular\_double, [973](#)
  - faxpy\_1\_modular\_double, [974](#)
  - fdot\_1\_modular\_double, [974](#)
  - fequal\_1\_modular\_double, [973](#)
  - fiszero\_1\_modular\_double, [973](#)
  - fneg\_1\_modular\_double, [972](#)
  - fnegin\_1\_modular\_double, [972](#)
  - freduce\_1\_modular\_double, [972](#)
  - freducein\_1\_modular\_double, [972](#)
  - fscal\_1\_modular\_double, [973](#)
  - fscaln\_1\_modular\_double, [973](#)
  - fsub\_1\_modular\_double, [974](#)
  - fsubin\_1\_modular\_double, [975](#)
  - fswap\_1\_modular\_double, [974](#)
  - fzero\_1\_modular\_double, [973](#)
- fflas\_lvl2.C, [975](#)
  - fadd\_2\_modular\_double, [979](#)
  - faddin\_2\_modular\_double, [979](#)
  - fassign\_2\_modular\_double, [976](#)
  - faxpy\_2\_modular\_double, [978](#)
  - fequal\_2\_modular\_double, [977](#)
  - fgemv\_2\_modular\_double, [980](#)
  - fger\_2\_modular\_double, [980](#)
  - fidentity\_2\_modular\_double, [977](#)
  - fiszero\_2\_modular\_double, [977](#)
  - fmove\_2\_modular\_double, [979](#)
  - fneg\_2\_modular\_double, [978](#)
  - fnegin\_2\_modular\_double, [978](#)
  - freduce\_2\_modular\_double, [977](#)
  - freducein\_2\_modular\_double, [977](#)
  - fscal\_2\_modular\_double, [978](#)
  - fscaln\_2\_modular\_double, [978](#)
  - fsub\_2\_modular\_double, [979](#)
  - fsubin\_2\_modular\_double, [979](#)
  - ftsv\_2\_modular\_double, [980](#)
  - fzero\_2\_modular\_double, [976](#)
- fflas\_lvl3.C, [981](#)
  - fgemm\_3\_modular\_double, [982](#)
  - fsquare\_3\_modular\_double, [982](#)
  - ftmm\_3\_modular\_double, [981](#)
  - ftsm\_3\_modular\_double, [981](#)
- fflas\_memory.h, [1043](#)
- fflas\_new
  - FFLAS, [157](#), [158](#), [192](#)
- FFLAS\_ORDER
  - FFLAS, [74](#)
- fflas\_pfgemm.inl, [869](#)
  - \_\_FFLASFFPACK\_DIMKPENALTY, [869](#)
  - \_\_FFLASFFPACK\_SEQPARTHRESHOLD, [869](#)
  - \_\_FFLASFFPACK\_fflas\_pfgemm\_INL, [869](#)
- fflas\_pftsm.inl, [870](#)
  - \_\_FFLASFFPACK\_fflas\_pftsm\_INL, [870](#)
  - PTRSM\_HYBRID\_THRESHOLD, [870](#)
- fflas\_plevel1.h, [1030](#)
- fflas\_randommatrix.h, [1043](#)
- FFLAS\_SIDE
  - FFLAS, [75](#)
- fflas\_simd.h, [870](#)
  - CONST, [871](#)
  - FLOAT\_MOD, [872](#)
  - INLINE, [871](#)
  - NORML\_MOD, [871](#)
  - PURE, [871](#)
  - Simd, [872](#)
  - SIMD\_INT, [871](#)
- fflas\_sparse.C, [982](#)
- fflas\_sparse.h, [880](#)
  - \_\_FFLASFFPACK\_CACHE\_LINE\_SIZE, [884](#)
  - assume\_aligned, [884](#)
  - DENSE\_THRESHOLD, [884](#)
  - index\_t, [884](#)
  - ROUND\_DOWN, [884](#)
- fflas\_sparse.inl, [884](#)
  - \_\_FFLASFFPACK\_fflas\_fflas\_sparse\_INL, [886](#)
- FFLAS\_TRANSPOSE
  - FFLAS, [74](#)
- fflas\_transpose.h, [912](#)
  - FFLAS\_TRANSPOSE\_BLOCKSIZE, [913](#)
  - LD, [913](#)
  - ST, [913](#)
- FFLAS\_TRANSPOSE\_BLOCKSIZE
  - fflas\_transpose.h, [913](#)
- FFLAS\_UPLO
  - FFLAS, [75](#)
- FflasAuto
  - FFLAS, [77](#)
- FflasBinary
  - FFLAS, [77](#)
- FflasColMajor
  - FFLAS, [74](#)
  - fflas\_c.h, [951](#)
  - ffpack\_c.h, [1006](#)
- FflasDense
  - FFLAS, [77](#)
- FflasDouble
  - FFLAS, [76](#)
  - fflas\_c.h, [952](#)
- FFLASFFPACK\_abort
  - debug.h, [1042](#)
- FFLASFFPACK\_check
  - debug.h, [1041](#)
- FFLASFFPACK\_checkers\_fflas\_inl\_H
  - checkers\_fflas.inl, [815](#)
- FFLASFFPACK\_checkers\_ffpack\_inl\_H
  - checkers\_ffpack.inl, [816](#)
- FFLASFFPACK\_COPY\_REDUCE

- fflas\_freduce.inl, [848](#)
- FFLASFFPACK\_PERM\_BKSIZE
  - ffpack\_permutation.inl, [939](#)
- FflasFloat
  - FFLAS, [76](#)
  - fflas\_c.h, [952](#)
- FflasGeneric
  - FFLAS, [76](#)
  - fflas\_c.h, [952](#)
- FflasLeft
  - FFLAS, [75](#)
  - fflas\_c.h, [952](#)
  - ffpack\_c.h, [1007](#)
- FflasLeftTri
  - FFLAS, [75](#)
- FflasLower
  - FFLAS, [75](#)
  - fflas\_c.h, [952](#)
  - ffpack\_c.h, [1007](#)
- FflasMaple
  - FFLAS, [77](#)
- FflasMath
  - FFLAS, [77](#)
- FflasNonUnit
  - FFLAS, [75](#)
  - fflas\_c.h, [952](#)
  - ffpack\_c.h, [1007](#)
- FflasNoTrans
  - FFLAS, [75](#)
  - fflas\_c.h, [952](#)
  - ffpack\_c.h, [1006](#)
- FflasRight
  - FFLAS, [75](#)
  - fflas\_c.h, [952](#)
  - ffpack\_c.h, [1007](#)
- FflasRightTri
  - FFLAS, [75](#)
- FflasRowMajor
  - FFLAS, [74](#)
  - fflas\_c.h, [951](#)
  - ffpack\_c.h, [1006](#)
- FflasSageMath
  - FFLAS, [77](#)
- FflasSMS
  - FFLAS, [77](#)
- FflasTrans
  - FFLAS, [75](#)
  - fflas\_c.h, [952](#)
  - ffpack\_c.h, [1006](#)
- FflasUnit
  - FFLAS, [75](#)
  - fflas\_c.h, [952](#)
  - ffpack\_c.h, [1007](#)
- FflasUpper
  - FFLAS, [75](#)
  - fflas\_c.h, [952](#)
  - ffpack\_c.h, [1007](#)
- FFPACK, [43](#), [291](#)
- \_PLUQ, [368](#)
- applyP, [309](#), [310](#), [370](#)
- applyP\_block, [362](#)
- Bruhat2EchelonPermutation, [350](#)
- buildMatrix, [356](#)
- CharPoly, [327](#), [328](#), [356](#), [374](#), [375](#)
- Checker\_charpoly, [308](#)
- Checker\_Det, [307](#)
- Checker\_invert, [308](#)
- Checker\_PLUQ, [307](#)
- chooseField, [395](#)
- chooseField< Givaro::ZRing< double > >, [395](#)
- chooseField< Givaro::ZRing< float > >, [395](#)
- chooseField< Givaro::ZRing< int32\_t > >, [395](#)
- chooseField< Givaro::ZRing< int64\_t > >, [395](#)
- ColRankProfileSubmatrix, [340](#), [379](#)
- ColRankProfileSubmatrixIndices, [339](#), [379](#)
- ColumnEchelonForm, [321](#), [373](#)
- ColumnRankProfile, [336](#), [337](#), [378](#)
- composePermutationsLLL, [365](#)
- composePermutationsLLM, [365](#)
- composePermutationsMLM, [366](#)
- CompressToBlockBiDiagonal, [349](#)
- ComputeRPermutation, [351](#), [354](#)
- cyclic\_shift\_col, [367](#), [370](#)
- cyclic\_shift\_mathPerm, [366](#)
- cyclic\_shift\_row, [366](#), [369](#)
- cyclic\_shift\_row\_col, [366](#), [369](#)
- Danilevski, [356](#)
- Det, [332](#), [333](#), [356](#), [357](#), [376](#), [377](#)
- doApplyS, [362](#)
- doApplyT, [363](#)
- ExpandBlockBiDiagonalToBruhat, [350](#)
- expandLCRE, [354](#)
- failure, [382](#)
- fflas\_const\_cast, [369](#), [382](#)
- fgesv, [312](#), [313](#), [371](#)
- fgetrs, [311](#), [312](#), [370](#)
- ForceCheck\_charpoly, [308](#)
- ForceCheck\_Det, [308](#)
- ForceCheck\_invert, [308](#)
- ForceCheck\_PLUQ, [308](#)
- fsytrf, [316](#), [317](#)
- fsytrf\_BC\_Crout, [357](#)
- fsytrf\_BC\_RL, [357](#)
- fsytrf\_LOW\_RPM\_BC\_Crout, [358](#)
- fsytrf\_nonunit, [317](#), [358](#), [359](#)
- fsytrf\_RPM, [359](#)
- fsytrf\_UP\_RPM, [358](#)
- fsytrf\_UP\_RPM\_BC\_Crout, [358](#)
- fsytrf\_UP\_RPM\_BC\_RL, [357](#)
- ftssyr2k, [316](#)
- ftstr, [315](#)
- fttri, [314](#), [371](#)
- fttrm, [315](#), [372](#)
- getEchelonForm, [342](#), [343](#)
- getEchelonForm< FFLAS\_FIELD< FFLAS\_ELT > >, [380](#)

- getEchelonTransform, [344](#)
- getEchelonTransform< FFLAS\_FIELD< FFLAS\_ELT  
> >, [380](#)
- getLTBruhatGen, [347](#), [348](#)
- getReducedEchelonForm, [344](#), [345](#)
- getReducedEchelonForm< FFLAS\_FIELD<  
FFLAS\_ELT > >, [381](#)
- getReducedEchelonTransform, [346](#)
- getReducedEchelonTransform< FFLAS\_FIELD<  
FFLAS\_ELT > >, [381](#)
- getTriangular, [341](#), [342](#)
- getTriangular< FFLAS\_FIELD< FFLAS\_ELT > >,  
[379](#), [380](#)
- getTridiagonal, [359](#)
- Invert, [325](#), [326](#), [374](#)
- Invert2, [326](#), [374](#)
- isOdd, [382](#)
- IsSingular, [331](#), [376](#)
- KrylovElim, [376](#)
- LAPACKPerm2MathPerm, [308](#)
- LeadingSubmatrixRankProfiles, [338](#)
- LQUPtoInverseOffFullRankMinor, [352](#), [382](#)
- LTBruhatGen, [347](#)
- LTQSorter, [349](#)
- LUdivine, [320](#), [360](#), [372](#)
- LUdivine\_gauss, [359](#), [373](#)
- LUdivine\_small, [360](#), [372](#)
- MathPerm2LAPACKPerm, [308](#)
- MatrixApplyS, [362](#), [363](#)
- MatrixApplyT, [364](#)
- MatVecMinPoly, [330](#), [375](#)
- MinPoly, [329](#), [375](#)
- MonotonicApplyP, [310](#)
- MonotonicCompress, [361](#)
- MonotonicCompressCycles, [361](#)
- MonotonicCompressMorePivots, [361](#)
- MonotonicExpand, [362](#)
- NonZeroRandomMatrix, [382](#), [383](#)
- NullSpaceBasis, [334](#), [378](#)
- pColumnEchelonForm, [321](#)
- pColumnRankProfile, [337](#)
- pDet, [332](#)
- PermApplyS, [363](#)
- PermApplyT, [365](#)
- PLUQ, [318](#), [319](#), [368](#), [369](#), [372](#)
- PLUQ\_basecaseCrout, [367](#)
- PLUQ\_basecaseV2, [367](#)
- PLUQ\_basecaseV3, [367](#)
- PLUQtoEchelonPermutation, [347](#)
- pPLUQ, [319](#)
- pRank, [331](#)
- pReducedColumnEchelonForm, [324](#)
- pReducedRowEchelonForm, [325](#)
- productBruhatxTS, [351](#), [354](#)
- pRowEchelonForm, [322](#)
- pRowRankProfile, [336](#)
- pSolve, [334](#)
- RandInt, [386](#)
- RandomIndexSubset, [388](#)
- RandomLTQSMatrixWithRankandQSorter, [395](#)
- RandomLTQSRankProfileMatrix, [389](#)
- RandomMatrix, [384](#)
- RandomMatrixWithDet, [394](#)
- RandomMatrixWithRank, [386](#), [387](#)
- RandomMatrixWithRankandRandomRPM, [392](#)
- RandomMatrixWithRankandRPM, [389](#), [390](#)
- RandomNullSpaceVector, [334](#), [352](#), [378](#)
- RandomPermutation, [388](#)
- RandomRankProfileMatrix, [388](#)
- RandomSymmetricMatrix, [386](#)
- RandomSymmetricMatrixWithRankandRandom-  
RPM, [393](#)
- RandomSymmetricMatrixWithRankandRPM, [390](#),  
[391](#)
- RandomSymmetricRankProfileMatrix, [389](#)
- RandomTriangularMatrix, [385](#)
- Rank, [330](#), [331](#), [376](#)
- RankProfileFromLU, [337](#)
- ReducedColumnEchelonForm, [323](#), [324](#), [374](#)
- ReducedRowEchelonForm, [324](#), [325](#), [373](#)
- RowEchelonForm, [322](#), [323](#), [373](#)
- RowRankProfile, [335](#), [336](#), [378](#)
- RowRankProfileSubmatrix, [340](#), [379](#)
- RowRankProfileSubmatrixIndices, [338](#), [379](#)
- Solve, [333](#), [377](#)
- solveLB, [353](#), [377](#)
- solveLB2, [353](#), [377](#)
- SpecRankProfile, [376](#)
- swapval, [389](#)
- threads\_fgemm, [368](#)
- threads\_ftsm, [368](#)
- TInverter, [351](#), [353](#)
- trinv\_left, [314](#), [372](#)
- ffpack-fgesv.C, [1105](#)
- main, [1105](#)
- ffpack-solve.C, [1105](#)
- main, [1106](#)
- ffpack.C, [983](#)
- applyP\_modular\_double, [988](#)
- ColRankProfileSubmatrix\_modular\_double, [1000](#)
- ColRankProfileSubmatrixIndices\_modular\_double,  
[1000](#)
- ColumnEchelonForm\_modular\_double, [991](#)
- ColumnEchelonForm\_modular\_float, [991](#)
- ColumnEchelonForm\_modular\_int32\_t, [992](#)
- ColumnRankProfile\_modular\_double, [999](#)
- composePermutationsLLL, [987](#)
- composePermutationsLLM, [987](#)
- composePermutationsMLM, [987](#)
- cyclic\_shift\_col\_modular\_double, [988](#)
- cyclic\_shift\_mathPerm, [988](#)
- cyclic\_shift\_row\_modular\_double, [988](#)
- Det\_modular\_double, [998](#)
- fgesv\_modular\_double, [989](#)
- fgesvin\_modular\_double, [989](#)
- fgetrsin\_modular\_double, [988](#)

- fgetrsv\_modular\_double, 989
- fttri\_modular\_double, 989
- fttrm\_modular\_double, 990
- getEchelonForm\_modular\_double, 1001
- getEchelonFormin\_modular\_double, 1001
- getEchelonTransform\_modular\_double, 1002
- getReducedEchelonForm\_modular\_double, 1002
- getReducedEchelonFormin\_modular\_double, 1002
- getReducedEchelonTransform\_modular\_double, 1002
- getTriangular\_modular\_double, 1001
- getTriangularin\_modular\_double, 1001
- Invert2\_modular\_double, 997
- Invert\_modular\_double, 996
- Invertin\_modular\_double, 996
- IsSingular\_modular\_double, 997
- KrylovElim\_modular\_double, 997
- LAPACKPerm2MathPerm, 986
- LeadingSubmatrixRankProfiles, 999
- LUdivine\_modular\_double, 990
- MathPerm2LAPACKPerm, 986
- MatrixApplyS\_modular\_double, 986
- MatrixApplyT\_modular\_double, 987
- NullSpaceBasis\_modular\_double, 999
- pColumnEchelonForm\_modular\_double, 993
- pColumnEchelonForm\_modular\_float, 994
- pColumnEchelonForm\_modular\_int32\_t, 995
- PermApplyS\_double, 987
- PermApplyT\_double, 987
- PLUQ\_modular\_double, 990
- PLUQtoEchelonPermutation, 1003
- pReducedColumnEchelonForm\_modular\_double, 994
- pReducedColumnEchelonForm\_modular\_float, 995
- pReducedColumnEchelonForm\_modular\_int32\_t, 996
- pReducedRowEchelonForm\_modular\_double, 994
- pReducedRowEchelonForm\_modular\_float, 995
- pReducedRowEchelonForm\_modular\_int32\_t, 996
- pRowEchelonForm\_modular\_double, 994
- pRowEchelonForm\_modular\_float, 995
- pRowEchelonForm\_modular\_int32\_t, 995
- RandomNullSpaceVector\_modular\_double, 998
- Rank\_modular\_double, 997
- RankProfileFromLU, 999
- ReducedColumnEchelonForm\_modular\_double, 991
- ReducedColumnEchelonForm\_modular\_float, 992
- ReducedColumnEchelonForm\_modular\_int32\_t, 993
- ReducedRowEchelonForm\_modular\_double, 991
- ReducedRowEchelonForm\_modular\_float, 992
- ReducedRowEchelonForm\_modular\_int32\_t, 993
- RowEchelonForm\_modular\_double, 991
- RowEchelonForm\_modular\_float, 992
- RowEchelonForm\_modular\_int32\_t, 993
- RowRankProfile\_modular\_double, 999
- RowRankProfileSubmatrix\_modular\_double, 1000
- RowRankProfileSubmatrixIndices\_modular\_double, 1000
- Solve\_modular\_double, 998
- solveLB2\_modular\_double, 998
- solveLB\_modular\_double, 998
- SpecRankProfile\_modular\_double, 997
- trinv\_left\_modular\_double, 990
- ffpack.dox, 913
- ffpack.h, 913
  - \_\_FFLASFFPACK\_FTRSSYR2K\_THRESHOLD, 922
  - \_\_FFLASFFPACK\_FTRSTR\_THRESHOLD, 922
- ffpack.inl, 922
  - \_\_FFLASFFPACK\_ffpack\_INL, 923
- FFPACK::Protected, 396
  - ArithProg, 400
  - CompressRows, 401
  - CompressRowsQA, 402
  - CompressRowsQK, 402
  - Danilevski, 399
  - DeCompressRows, 402
  - DeCompressRowsQA, 402
  - DeCompressRowsQK, 402
  - fgemv\_kgf, 399
  - GaussJordan, 398
  - Hybrid\_KGF\_LUK\_MinPoly, 401
  - KellerGehrig, 398
  - KGFast, 398
  - KGFast\_generalized, 399
  - LUdivine\_construct, 397, 403
  - LUKrylov, 399
  - LUKrylov\_KGFast, 400
  - MatVecMinPoly, 400
  - newD, 401
  - RandomKrylovPrecond, 400
  - updateD, 401
- ffpack\_bruhatgen.inl, 923
  - \_\_FFLASFFPACK\_ffpack\_bruhatgen\_inl, 925
- ffpack\_c.h, 1003
  - applyP\_modular\_double, 1010
  - ColRankProfileSubmatrix\_modular\_double, 1020
  - ColRankProfileSubmatrixIndices\_modular\_double, 1020
  - ColumnEchelonForm\_modular\_double, 1013
  - ColumnEchelonForm\_modular\_float, 1013
  - ColumnEchelonForm\_modular\_int32\_t, 1013
  - ColumnRankProfile\_modular\_double, 1019
  - composePermutationsLLL, 1009
  - composePermutationsLLM, 1009
  - composePermutationsMLM, 1009
  - cyclic\_shift\_col\_modular\_double, 1009
  - cyclic\_shift\_mathPerm, 1009
  - cyclic\_shift\_row\_modular\_double, 1009
  - Det\_modular\_double, 1017
  - FFLAS\_C\_DIAG, 1007
  - FFLAS\_C\_ORDER, 1006

- FFLAS\_C\_SIDE, [1007](#)
- FFLAS\_C\_TRANSPOSE, [1006](#)
- FFLAS\_C\_UPLO, [1006](#)
- FflasColMajor, [1006](#)
- FflasLeft, [1007](#)
- FflasLower, [1007](#)
- FflasNonUnit, [1007](#)
- FflasNoTrans, [1006](#)
- FflasRight, [1007](#)
- FflasRowMajor, [1006](#)
- FflasTrans, [1006](#)
- FflasUnit, [1007](#)
- FflasUpper, [1007](#)
- FFPACK\_C\_CHARPOLY\_TAG, [1007](#)
- FFPACK\_C\_LU\_TAG, [1007](#)
- FFPACK\_C\_MINPOLY\_TAG, [1007](#)
- FFPACK\_COMPILED, [1006](#)
- FfpackArithProg, [1007](#)
- FfpackDanilevski, [1007](#)
- FfpackDense, [1008](#)
- FfpackHybrid, [1007](#)
- FfpackKG, [1007](#)
- FfpackKGF, [1008](#)
- FfpackKGFast, [1007](#)
- FfpackKGFastG, [1007](#)
- FfpackLUK, [1007](#)
- FfpackSingular, [1007](#)
- FfpackSlabRecursive, [1007](#)
- FfpackTileRecursive, [1007](#)
- fgesv\_modular\_double, [1011](#)
- fgesvin\_modular\_double, [1010](#)
- fgetrs\_modular\_double, [1010](#)
- fgetrsin\_modular\_double, [1010](#)
- fttrtri\_modular\_double, [1011](#)
- fttrrm\_modular\_double, [1011](#)
- getEchelonForm\_modular\_double, [1021](#)
- getEchelonFormin\_modular\_double, [1021](#)
- getEchelonTransform\_modular\_double, [1021](#)
- getReducedEchelonForm\_modular\_double, [1022](#)
- getReducedEchelonFormin\_modular\_double, [1022](#)
- getReducedEchelonTransform\_modular\_double, [1022](#)
- getTriangular\_modular\_double, [1020](#)
- getTriangularin\_modular\_double, [1020](#)
- Invert2\_modular\_double, [1016](#)
- Invert\_modular\_double, [1016](#)
- Invertin\_modular\_double, [1016](#)
- IsSingular\_modular\_double, [1017](#)
- KrylovElim\_modular\_double, [1016](#)
- LAPACKPerm2MathPerm, [1008](#)
- LeadingSubmatrixRankProfiles, [1019](#)
- LUdivine\_gauss\_modular\_double, [1012](#)
- LUdivine\_modular\_double, [1012](#)
- LUdivine\_small\_modular\_double, [1012](#)
- MathPerm2LAPACKPerm, [1008](#)
- MatrixApplyS\_modular\_double, [1008](#)
- MatrixApplyT\_modular\_double, [1008](#)
- NullSpaceBasis\_modular\_double, [1018](#)
- PermApplyS\_double, [1008](#)
- PermApplyT\_double, [1009](#)
- PLUQ\_modular\_double, [1012](#)
- PLUQtoEchelonPermutation, [1022](#)
- RandomNullSpaceVector\_modular\_double, [1018](#)
- Rank\_modular\_double, [1017](#)
- RankProfileFromLU, [1019](#)
- ReducedColumnEchelonForm\_modular\_double, [1014](#)
- ReducedColumnEchelonForm\_modular\_float, [1014](#)
- ReducedColumnEchelonForm\_modular\_int32\_t, [1015](#)
- ReducedRowEchelonForm2\_modular\_double, [1015](#)
- ReducedRowEchelonForm\_modular\_double, [1014](#)
- ReducedRowEchelonForm\_modular\_float, [1015](#)
- ReducedRowEchelonForm\_modular\_int32\_t, [1015](#)
- REF\_modular\_double, [1016](#)
- RowEchelonForm\_modular\_double, [1013](#)
- RowEchelonForm\_modular\_float, [1013](#)
- RowEchelonForm\_modular\_int32\_t, [1014](#)
- RowRankProfile\_modular\_double, [1019](#)
- RowRankProfileSubmatrix\_modular\_double, [1020](#)
- RowRankProfileSubmatrixIndices\_modular\_double, [1019](#)
- Solve\_modular\_double, [1017](#)
- solveLB2\_modular\_double, [1018](#)
- solveLB\_modular\_double, [1018](#)
- SpecRankProfile\_modular\_double, [1017](#)
- trinv\_left\_modular\_double, [1011](#)
- FFPACK\_C\_CHARPOLY\_TAG
  - ffpack\_c.h, [1007](#)
- FFPACK\_C\_LU\_TAG
  - ffpack\_c.h, [1007](#)
- FFPACK\_C\_MINPOLY\_TAG
  - ffpack\_c.h, [1007](#)
- ffpack\_charpoly.inl, [925](#)
- \_\_FFLASFFPACK\_charpoly\_INL, [925](#)
- ffpack\_charpoly\_danilevski.inl, [925](#)
- \_\_FFLASFFPACK\_ffpack\_charpoly\_danilveski\_INL, [926](#)
- ffpack\_charpoly\_kgfast.inl, [926](#)
- \_\_FFLASFFPACK\_ffpack\_charpoly\_kgfast\_INL, [926](#)
- ffpack\_charpoly\_kgfastgeneralized.inl, [926](#)
- \_\_FFLASFFPACK\_ffpack\_charpoly\_kgfastgeneralized\_INL, [927](#)
- ffpack\_charpoly\_kglu.inl, [927](#)
- \_\_FFLASFFPACK\_ffpack\_charpoly\_kglu\_INL, [927](#)
- ffpack\_charpoly\_mp.inl, [927](#)
- \_\_FFPACK\_charpoly\_mp\_INL, [928](#)
- FFPACK\_COMPILED
  - ffpack\_c.h, [1006](#)
- ffpack\_det\_mp.inl, [928](#)
- \_\_FFPACK\_det\_mp\_INL, [928](#)
- ffpack\_echelonforms.inl, [929](#)

- \_\_FFLASFFPACK\_GAUSSJORDAN\_BASECASE, 930
- \_\_FFLASFFPACK\_ffpack\_echelon\_forms\_INL, 930
- ffpack\_fgesv.inl, 930
- \_\_FFLASFFPACK\_ffpack\_fgesv\_INL, 930
- ffpack\_fgetrs.inl, 930
- \_\_FFLASFFPACK\_ffpack\_fgetrs\_INL, 931
- ffpack\_frobenius.inl, 931
- ffpack\_fsytrf.inl, 932
- \_\_FFLASFFPACK\_ffpack\_fsytrf\_INL, 933
- ffpack\_ftrssyr2k.inl, 933
- \_\_FFLASFFPACK\_ffpack\_ftrssyr2k\_INL, 933
- ffpack\_ftrstr.inl, 933
- \_\_FFLASFFPACK\_ffpack\_ftrstr\_INL, 934
- ffpack\_ftrtr.inl, 934
- \_\_FFLASFFPACK\_ffpack\_ftrtr\_INL, 934
- ENABLE\_ALL\_CHECKINGS, 934
- ffpack\_inst.C, 1023
- \_\_FFPACK\_INST\_C, 1023
- FFLAS\_COMPILED, 1023
- FFLAS\_ELT, 1023, 1024
- FFLAS\_FIELD, 1023
- INST\_OR\_DECL, 1023
- ffpack\_inst.h, 1024
- FFLAS\_COMPILED, 1024
- FFLAS\_ELT, 1024, 1025
- FFLAS\_FIELD, 1024
- INST\_OR\_DECL, 1024
- ffpack\_inst\_implem.inl, 1025
- ffpack\_invert.inl, 934
- \_\_FFLASFFPACK\_ffpack\_invert\_INL, 935
- ffpack\_krylovelim.inl, 935
- \_\_FFLASFFPACK\_ffpack\_krylovelim\_INL, 935
- ffpack\_ludivine.inl, 935
- \_\_FFLASFFPACK\_ffpack\_ludivine\_INL, 936
- ffpack\_ludivine\_mp.inl, 936
- \_\_FFPACK\_ludivine\_mp\_INL, 936
- ffpack\_minpoly.inl, 937
- \_\_FFLASFFPACK\_ffpack\_minpoly\_INL, 937
- ffpack\_permutation.inl, 937
- \_\_FFLASFFPACK\_ffpack\_permutation\_INL, 939
- FFLASFFPACK\_PERM\_BKSIZE, 939
- ffpack\_pluq.inl, 940
- \_\_FFLASFFPACK\_ffpack\_pluq\_INL, 940
- CROUT, 940
- ffpack\_pluq\_mp.inl, 940
- \_\_FFPACK\_pluq\_mp\_INL, 941
- ffpack\_ppluq.inl, 941
- \_\_FFLASFFPACK\_ffpack\_ppluq\_INL, 941
- \_\_FFLAS\_TRSM\_READONLY, 941
- PBASECASE\_K, 941
- ffpack\_rankprofiles.inl, 942
- \_\_FFLASFFPACK\_ffpack\_rank\_profiles\_INL, 943
- FfpackArithProg
- ffpack\_c.h, 1007
- FfpackDanilevski
- ffpack\_c.h, 1007
- FfpackDense
- ffpack\_c.h, 1008
- FfpackHybrid
- ffpack\_c.h, 1007
- FfpackKG
- ffpack\_c.h, 1007
- FfpackKGF
- ffpack\_c.h, 1008
- FfpackKGFast
- ffpack\_c.h, 1007
- FfpackKGFastG
- ffpack\_c.h, 1007
- FfpackLUK
- ffpack\_c.h, 1007
- FfpackSingular
- ffpack\_c.h, 1007
- FfpackSlabRecursive
- ffpack\_c.h, 1007
- FfpackTileRecursive
- ffpack\_c.h, 1007
- fgemm
- FFLAS, 89–91, 93–98, 146, 179, 180
- fgemm\_3\_modular\_double
- fflas\_c.h, 960
- fflas\_lvl3.C, 982
- fgemm\_classical.inl, 835
- fgemm\_classical\_mp.inl, 835
- \_\_FFPACK\_fgemm\_classical\_INL, 837
- fgemm\_convert
- FFLAS::Protected, 217
- fgemm\_winograd.inl, 837
- \_\_FFLASFFPACK\_fflas\_fflas\_fgemm\_winograd\_INL, 838
- NEWWINO, 838
- fgemv
- FFLAS, 98–103, 175, 187
- fgemv\_2\_modular\_double
- fflas\_c.h, 959
- fflas\_lvl2.C, 980
- fgemv\_convert
- FFLAS::Protected, 222
- fgemv\_kgf
- FFPACK::Protected, 399
- fger
- FFLAS, 104–107, 176
- fger\_2\_modular\_double
- fflas\_c.h, 959
- fflas\_lvl2.C, 980
- fger\_convert
- FFLAS::Protected, 223
- fgesv
- FFPACK, 312, 313, 371
- fgesv\_modular\_double
- ffpack.C, 989
- ffpack\_c.h, 1011
- fgesvin\_modular\_double
- ffpack.C, 989
- ffpack\_c.h, 1010



- fgetrs
  - FFPACK, [311](#), [312](#), [370](#)
- fgetrs\_modular\_double
  - ffpack\_c.h, [1010](#)
- fgetrsin\_modular\_double
  - ffpack.C, [988](#)
  - ffpack\_c.h, [1010](#)
- fgetrsv\_modular\_double
  - ffpack.C, [989](#)
- fidentity
  - FFLAS, [139](#), [168](#)
- fidentity\_2\_modular\_double
  - fflas\_c.h, [956](#)
  - fflas\_lvl2.C, [977](#)
- Field
  - Bench< Elt >, [411](#)
  - benchmark-fgemm-rns.C, [785](#)
  - benchmark-pluq.C, [798](#)
  - FieldSimd< \_Field >, [451](#)
  - Sparse< \_Field, SparseMatrix\_t::COO >, [734](#)
  - Sparse< \_Field, SparseMatrix\_t::COO\_ZO >, [735](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR >, [737](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR\_HYB >, [739](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR\_ZO >, [741](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL >, [742](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_ZO >, [747](#)
  - Sparse< \_Field, SparseMatrix\_t::HYB\_ZO >, [749](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL >, [750](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, [752](#)
  - Test< Elt >, [760](#)
  - test-compressQ.C, [1054](#)
- field
  - associatedDelayedField< const FFPACK::RNSIntegerMod< T > >, [408](#)
  - associatedDelayedField< const Givaro::Modular< T, X > >, [409](#)
  - associatedDelayedField< const Givaro::ModularBalanced< T > >, [409](#)
  - associatedDelayedField< const Givaro::ZRing< T > >, [410](#)
  - associatedDelayedField< Field >, [408](#)
- field-traits.h, [943](#)
- field.dox, [945](#)
- FieldMax
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [500](#)
- FieldMin
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [499](#)
- FieldSimd
  - FieldSimd< \_Field >, [451](#), [452](#)
- FieldSimd< \_Field >, [450](#)
  - add, [452](#)
  - add\_r, [452](#), [453](#)
  - addin, [452](#)
  - addin\_r, [453](#)
  - alignment, [456](#)
  - axpy, [454](#), [455](#)
  - axpy\_r, [455](#)
  - axpyin, [455](#)
  - axpyin\_r, [455](#)
  - Element, [451](#)
  - Field, [451](#)
  - FieldSimd, [451](#), [452](#)
  - init, [452](#)
  - maxpy, [455](#)
  - maxpyin, [456](#)
  - mod, [454](#)
  - mul, [454](#)
  - mul\_r, [454](#)
  - mulin, [454](#)
  - operator=, [452](#)
  - scalar\_t, [451](#)
  - simd, [451](#)
  - sub, [453](#)
  - sub\_r, [453](#)
  - subin, [453](#)
  - subin\_r, [453](#)
  - vect\_size, [456](#)
  - vect\_t, [451](#)
  - zero, [454](#)
- FieldTraits< FFPACK::RNSInteger< T > >, [457](#)
  - balanced, [457](#)
  - category, [457](#)
- FieldTraits< FFPACK::RNSIntegerMod< T > >, [457](#)
  - balanced, [458](#)
  - category, [457](#)
- FieldTraits< Field >, [456](#)
  - balanced, [457](#)
  - category, [456](#)
- FieldTraits< Givaro::Modular< Element > >, [458](#)
  - balanced, [458](#)
  - category, [458](#)
- FieldTraits< Givaro::ModularBalanced< Element > >, [458](#)
  - balanced, [459](#)
  - category, [458](#)
- FieldTraits< Givaro::ZRing< double > >, [459](#)
  - balanced, [459](#)
  - category, [459](#)
- FieldTraits< Givaro::ZRing< float > >, [459](#)
  - balanced, [460](#)
  - category, [459](#)
- FieldTraits< Givaro::ZRing< Givaro::Integer > >, [460](#)
  - balanced, [460](#)
  - category, [460](#)
- FieldTraits< Givaro::ZRing< int16\_t > >, [460](#)
  - balanced, [461](#)
  - category, [460](#)
- FieldTraits< Givaro::ZRing< int32\_t > >, [461](#)
  - balanced, [461](#)
  - category, [461](#)
- FieldTraits< Givaro::ZRing< int64\_t > >, [461](#)
  - balanced, [461](#)
  - category, [461](#)

- FieldTraits< Givaro::ZRing< RecInt::ruint< K > > >, 462
  - balanced, 462
  - category, 462
- FieldTraits< Givaro::ZRing< uint16\_t > >, 462
  - balanced, 462
  - category, 462
- FieldTraits< Givaro::ZRing< uint32\_t > >, 462
  - balanced, 463
  - category, 463
- FieldTraits< Givaro::ZRing< uint64\_t > >, 463
  - balanced, 463
  - category, 463
- fill\_value
  - benchmark-fgemv.C, 789
- findArgument
  - args-parser.h, 1039
- finit
  - FFLAS, 109, 112, 131, 139, 160, 170
- finit\_rns
  - FFLAS, 157, 158
- finit\_trans\_rns
  - FFLAS, 157
- first\_component
  - Compose< H1, H2 >, 428
- fiszero
  - FFLAS, 134, 138, 162, 168
- fiszero\_1\_modular\_double
  - fflas\_c.h, 953
  - fflas\_lvl1.C, 973
- fiszero\_2\_modular\_double
  - fflas\_c.h, 956
  - fflas\_lvl2.C, 977
- Fixed, 463
- FixedPreIntTag, 463
- flimits.h, 1045
  - in\_range, 1046
- FLOAT\_MOD
  - fflas\_simd.h, 872
- FloatingPointTestDistribution
  - ScalFunctionsBase< Element, typename enable\_if< is\_floating\_point< Element >::value >::type >::FloatingPointTestDistribution, 464
- Floats
  - benchmark-dgemm.C, 778
- floor
  - ScalFunctionsBase< Element, typename enable\_if< is\_floating\_point< Element >::value >::type >, 557
  - Simd256\_impl< true, false, true, 8 >, 628
  - Simd512\_impl< true, false, true, 8 >, 710
- fma
  - ScalFunctionsBase< Element, typename enable\_if< is\_floating\_point< Element >::value >::type >, 557
  - ScalFunctionsBase< Element, typename enable\_if< is\_integral< Element >::value >::type >, 559
- fmadd
  - ScalFunctionsBase< Element, typename enable\_if< is\_floating\_point< Element >::value >::type >, 557
  - ScalFunctionsBase< Element, typename enable\_if< is\_integral< Element >::value >::type >, 559
- ScalFunctions< Element >, 554
  - Simd128\_impl< true, true, false, 2 >, 570
  - Simd128\_impl< true, true, false, 4 >, 580
  - Simd128\_impl< true, true, false, 8 >, 590
  - Simd128\_impl< true, true, true, 2 >, 598
  - Simd128\_impl< true, true, true, 4 >, 607
  - Simd128\_impl< true, true, true, 8 >, 616
  - Simd256\_impl< true, false, true, 8 >, 627
  - Simd256\_impl< true, true, false, 2 >, 638
  - Simd256\_impl< true, true, false, 4 >, 655
  - Simd256\_impl< true, true, false, 8 >, 666
  - Simd256\_impl< true, true, true, 2 >, 674
  - Simd256\_impl< true, true, true, 4 >, 684, 691
  - Simd256\_impl< true, true, true, 8 >, 700
  - Simd512\_impl< true, false, true, 8 >, 708
  - Simd512\_impl< true, true, false, 8 >, 719
  - Simd512\_impl< true, true, true, 8 >, 728
- fmaddin
  - ScalFunctions< Element >, 554
    - Simd128\_impl< true, true, false, 2 >, 570
    - Simd128\_impl< true, true, false, 4 >, 580
    - Simd128\_impl< true, true, false, 8 >, 590
    - Simd128\_impl< true, true, true, 2 >, 598
    - Simd128\_impl< true, true, true, 4 >, 607
    - Simd128\_impl< true, true, true, 8 >, 616
    - Simd256\_impl< true, false, true, 8 >, 627
    - Simd256\_impl< true, true, false, 2 >, 638
    - Simd256\_impl< true, true, false, 4 >, 655
    - Simd256\_impl< true, true, false, 8 >, 666
    - Simd256\_impl< true, true, true, 2 >, 674
    - Simd256\_impl< true, true, true, 4 >, 684, 691
    - Simd256\_impl< true, true, true, 8 >, 700
    - Simd512\_impl< true, false, true, 8 >, 709
    - Simd512\_impl< true, true, false, 8 >, 719
    - Simd512\_impl< true, true, true, 8 >, 728
- fmaddx
  - ScalFunctionsBase< Element, typename enable\_if< is\_integral< Element >::value >::type >, 559
    - Simd128\_impl< true, true, false, 2 >, 566
    - Simd128\_impl< true, true, false, 4 >, 576
    - Simd128\_impl< true, true, false, 8 >, 586
    - Simd128\_impl< true, true, true, 2 >, 598
    - Simd128\_impl< true, true, true, 4 >, 607
    - Simd128\_impl< true, true, true, 8 >, 616
    - Simd256\_impl< true, true, false, 2 >, 633
    - Simd256\_impl< true, true, false, 4 >, 645, 648
    - Simd256\_impl< true, true, false, 8 >, 662
    - Simd256\_impl< true, true, true, 2 >, 674
    - Simd256\_impl< true, true, true, 4 >, 685, 691
    - Simd256\_impl< true, true, true, 8 >, 700
    - Simd512\_impl< true, true, false, 8 >, 715
    - Simd512\_impl< true, true, true, 8 >, 728
- fmaddxin
  - ScalFunctionsBase< Element, typename enable\_if< is\_integral< Element >::value >::type >, 559
    - Simd128\_impl< true, true, false, 2 >, 566
    - Simd128\_impl< true, true, false, 4 >, 576
    - Simd128\_impl< true, true, false, 8 >, 586



- Simd128\_impl< true, true, true, 2 >, [599](#)
- Simd128\_impl< true, true, true, 4 >, [607](#)
- Simd128\_impl< true, true, true, 8 >, [617](#)
- Simd256\_impl< true, true, false, 2 >, [634](#)
- Simd256\_impl< true, true, false, 4 >, [645](#), [648](#)
- Simd256\_impl< true, true, false, 8 >, [662](#)
- Simd256\_impl< true, true, true, 2 >, [674](#)
- Simd256\_impl< true, true, true, 4 >, [685](#), [691](#)
- Simd256\_impl< true, true, true, 8 >, [700](#)
- Simd512\_impl< true, true, false, 8 >, [715](#)
- Simd512\_impl< true, true, true, 8 >, [728](#)
- fmove
  - FFLAS, [142](#), [172](#)
- fmove\_2\_modular\_double
  - fflas\_c.h, [958](#)
  - fflas\_lvl2.C, [979](#)
- fmsub
  - ScalFunctions< Element >, [554](#)
  - Simd128\_impl< true, true, false, 2 >, [570](#)
  - Simd128\_impl< true, true, false, 4 >, [580](#)
  - Simd128\_impl< true, true, false, 8 >, [591](#)
  - Simd128\_impl< true, true, true, 2 >, [599](#)
  - Simd128\_impl< true, true, true, 4 >, [608](#)
  - Simd128\_impl< true, true, true, 8 >, [617](#)
  - Simd256\_impl< true, false, true, 8 >, [627](#)
  - Simd256\_impl< true, true, false, 2 >, [638](#)
  - Simd256\_impl< true, true, false, 4 >, [656](#)
  - Simd256\_impl< true, true, false, 8 >, [666](#)
  - Simd256\_impl< true, true, true, 2 >, [675](#)
  - Simd256\_impl< true, true, true, 4 >, [685](#), [692](#)
  - Simd256\_impl< true, true, true, 8 >, [701](#)
  - Simd512\_impl< true, false, true, 8 >, [709](#)
  - Simd512\_impl< true, true, false, 8 >, [720](#)
  - Simd512\_impl< true, true, true, 8 >, [729](#)
- fmsubbin
  - ScalFunctions< Element >, [554](#)
  - Simd128\_impl< true, true, false, 2 >, [571](#)
  - Simd128\_impl< true, true, false, 4 >, [581](#)
  - Simd128\_impl< true, true, false, 8 >, [591](#)
  - Simd128\_impl< true, true, true, 2 >, [599](#)
  - Simd128\_impl< true, true, true, 4 >, [608](#)
  - Simd128\_impl< true, true, true, 8 >, [617](#)
  - Simd256\_impl< true, false, true, 8 >, [627](#)
  - Simd256\_impl< true, true, false, 2 >, [638](#)
  - Simd256\_impl< true, true, false, 4 >, [656](#)
  - Simd256\_impl< true, true, false, 8 >, [666](#)
  - Simd256\_impl< true, true, true, 2 >, [675](#)
  - Simd256\_impl< true, true, true, 4 >, [685](#), [692](#)
  - Simd256\_impl< true, true, true, 8 >, [701](#)
  - Simd512\_impl< true, false, true, 8 >, [709](#)
  - Simd512\_impl< true, true, false, 8 >, [720](#)
  - Simd512\_impl< true, true, true, 8 >, [729](#)
- fmsubx
  - ScalFunctionsBase< Element, typename enable\_if<
    - is\_integral< Element >::value >::type >, [559](#)
 >
    - Simd128\_impl< true, true, false, 2 >, [566](#)
    - Simd128\_impl< true, true, false, 4 >, [577](#)
    - Simd128\_impl< true, true, false, 8 >, [587](#)
- Simd128\_impl< true, true, true, 2 >, [599](#)
- Simd128\_impl< true, true, true, 4 >, [608](#)
- Simd128\_impl< true, true, true, 8 >, [617](#)
- Simd256\_impl< true, true, false, 2 >, [634](#)
- Simd256\_impl< true, true, false, 4 >, [646](#), [648](#)
- Simd256\_impl< true, true, false, 8 >, [662](#)
- Simd256\_impl< true, true, true, 2 >, [675](#)
- Simd256\_impl< true, true, true, 4 >, [686](#), [692](#)
- Simd256\_impl< true, true, true, 8 >, [701](#)
- Simd512\_impl< true, true, false, 8 >, [715](#)
- Simd512\_impl< true, true, true, 8 >, [729](#)
- fmsubxin
  - ScalFunctionsBase< Element, typename enable\_if<
    - is\_integral< Element >::value >::type >, [560](#)
 >
    - Simd128\_impl< true, true, false, 2 >, [567](#)
    - Simd128\_impl< true, true, false, 4 >, [577](#)
    - Simd128\_impl< true, true, false, 8 >, [587](#)
    - Simd128\_impl< true, true, true, 2 >, [600](#)
    - Simd128\_impl< true, true, true, 4 >, [608](#)
    - Simd128\_impl< true, true, true, 8 >, [618](#)
    - Simd256\_impl< true, true, false, 2 >, [634](#)
    - Simd256\_impl< true, true, false, 4 >, [646](#), [648](#)
    - Simd256\_impl< true, true, false, 8 >, [662](#)
    - Simd256\_impl< true, true, true, 2 >, [675](#)
    - Simd256\_impl< true, true, true, 4 >, [686](#), [692](#)
    - Simd256\_impl< true, true, true, 8 >, [701](#)
    - Simd512\_impl< true, true, false, 8 >, [716](#)
    - Simd512\_impl< true, true, true, 8 >, [729](#)
- fneg
  - FFLAS, [132](#), [141](#), [161](#), [170](#)
- fneg\_1\_modular\_double
  - fflas\_c.h, [953](#)
  - fflas\_lvl1.C, [972](#)
- fneg\_2\_modular\_double
  - fflas\_c.h, [957](#)
  - fflas\_lvl2.C, [978](#)
- fnegin
  - FFLAS, [132](#), [140](#), [160](#), [170](#)
- fnegin\_1\_modular\_double
  - fflas\_c.h, [953](#)
  - fflas\_lvl1.C, [972](#)
- fnegin\_2\_modular\_double
  - fflas\_c.h, [957](#)
  - fflas\_lvl2.C, [978](#)
- fnmadd
  - ScalFunctions< Element >, [554](#)
  - Simd128\_impl< true, true, false, 2 >, [570](#)
  - Simd128\_impl< true, true, false, 4 >, [580](#)
  - Simd128\_impl< true, true, false, 8 >, [590](#)
  - Simd128\_impl< true, true, true, 2 >, [599](#)
  - Simd128\_impl< true, true, true, 4 >, [608](#)
  - Simd128\_impl< true, true, true, 8 >, [617](#)
  - Simd256\_impl< true, false, true, 8 >, [627](#)
  - Simd256\_impl< true, true, false, 2 >, [638](#)
  - Simd256\_impl< true, true, false, 4 >, [655](#)
  - Simd256\_impl< true, true, false, 8 >, [666](#)
  - Simd256\_impl< true, true, true, 2 >, [674](#)
  - Simd256\_impl< true, true, true, 4 >, [685](#), [691](#)

- Simd256\_impl< true, true, true, 8 >, [700](#)
- Simd512\_impl< true, false, true, 8 >, [709](#)
- Simd512\_impl< true, true, false, 8 >, [719](#)
- Simd512\_impl< true, true, true, 8 >, [728](#)
- fnmaddin
  - ScalFunctions< Element >, [554](#)
  - Simd128\_impl< true, true, false, 2 >, [570](#)
  - Simd128\_impl< true, true, false, 4 >, [580](#)
  - Simd128\_impl< true, true, false, 8 >, [590](#)
  - Simd128\_impl< true, true, true, 2 >, [599](#)
  - Simd128\_impl< true, true, true, 4 >, [608](#)
  - Simd128\_impl< true, true, true, 8 >, [617](#)
  - Simd256\_impl< true, false, true, 8 >, [627](#)
  - Simd256\_impl< true, true, false, 2 >, [638](#)
  - Simd256\_impl< true, true, false, 4 >, [655](#)
  - Simd256\_impl< true, true, false, 8 >, [666](#)
  - Simd256\_impl< true, true, true, 2 >, [674](#)
  - Simd256\_impl< true, true, true, 4 >, [685](#), [691](#)
  - Simd256\_impl< true, true, true, 8 >, [700](#)
  - Simd512\_impl< true, false, true, 8 >, [709](#)
  - Simd512\_impl< true, true, false, 8 >, [720](#)
  - Simd512\_impl< true, true, true, 8 >, [728](#)
- fnmaddx
  - ScalFunctionsBase< Element, typename enable\_if< is\_integral< Element >::value >::type >, [560](#)
  - Simd128\_impl< true, true, false, 2 >, [566](#)
  - Simd128\_impl< true, true, false, 4 >, [576](#)
  - Simd128\_impl< true, true, false, 8 >, [586](#)
  - Simd128\_impl< true, true, true, 2 >, [599](#)
  - Simd128\_impl< true, true, true, 4 >, [608](#)
  - Simd128\_impl< true, true, true, 8 >, [617](#)
  - Simd256\_impl< true, true, false, 2 >, [634](#)
  - Simd256\_impl< true, true, false, 4 >, [645](#), [648](#)
  - Simd256\_impl< true, true, false, 8 >, [662](#)
  - Simd256\_impl< true, true, true, 2 >, [674](#)
  - Simd256\_impl< true, true, true, 4 >, [685](#), [691](#)
  - Simd256\_impl< true, true, true, 8 >, [700](#)
  - Simd512\_impl< true, true, false, 8 >, [715](#)
  - Simd512\_impl< true, true, true, 8 >, [728](#)
- fnmaddxin
  - ScalFunctionsBase< Element, typename enable\_if< is\_integral< Element >::value >::type >, [560](#)
  - Simd128\_impl< true, true, false, 2 >, [566](#)
  - Simd128\_impl< true, true, false, 4 >, [576](#)
  - Simd128\_impl< true, true, false, 8 >, [587](#)
  - Simd128\_impl< true, true, true, 2 >, [599](#)
  - Simd128\_impl< true, true, true, 4 >, [608](#)
  - Simd128\_impl< true, true, true, 8 >, [617](#)
  - Simd256\_impl< true, true, false, 2 >, [634](#)
  - Simd256\_impl< true, true, false, 4 >, [646](#), [648](#)
  - Simd256\_impl< true, true, false, 8 >, [662](#)
  - Simd256\_impl< true, true, true, 2 >, [675](#)
  - Simd256\_impl< true, true, true, 4 >, [685](#), [691](#)
  - Simd256\_impl< true, true, true, 8 >, [700](#)
  - Simd512\_impl< true, true, false, 8 >, [715](#)
  - Simd512\_impl< true, true, true, 8 >, [728](#)
- FOR1D
  - parallel.h, [1033](#)
- FOR2D
  - parallel.h, [1033](#)
- FORBLOCK1D
  - parallel.h, [1032](#)
- FORBLOCK2D
  - parallel.h, [1033](#)
- ForceCheck\_charpoly
  - FFPACK, [308](#)
- ForceCheck\_Det
  - FFPACK, [308](#)
- ForceCheck\_fgemm
  - FFLAS, [72](#)
- ForceCheck\_ftrsm
  - FFLAS, [72](#)
- ForceCheck\_invert
  - FFPACK, [308](#)
- ForceCheck\_PLUQ
  - FFPACK, [308](#)
- frand
  - FFLAS, [133](#), [138](#)
- freduce
  - FFLAS, [108–110](#), [113](#), [159](#), [169](#)
  - FFLAS::details, [205](#), [206](#)
- freduce\_1\_modular\_double
  - fflas\_c.h, [953](#)
  - fflas\_lvl1.C, [972](#)
- freduce\_2\_modular\_double
  - fflas\_c.h, [957](#)
  - fflas\_lvl2.C, [977](#)
- freduce\_constoverride
  - FFLAS, [109](#), [112](#)
- freducein\_1\_modular\_double
  - fflas\_c.h, [953](#)
  - fflas\_lvl1.C, [972](#)
- freducein\_2\_modular\_double
  - fflas\_c.h, [957](#)
  - fflas\_lvl2.C, [977](#)
- freivalds
  - FFLAS, [113](#)
- fscal
  - FFLAS, [114–118](#), [163](#), [171](#)
  - FFLAS::details, [206](#), [207](#)
- fscal\_1\_modular\_double
  - fflas\_c.h, [954](#)
  - fflas\_lvl1.C, [973](#)
- fscal\_2\_modular\_double
  - fflas\_c.h, [957](#)
  - fflas\_lvl2.C, [978](#)
- fscalin
  - FFLAS, [114–118](#), [163](#), [171](#)
  - FFLAS::details, [206](#), [207](#)
- fscalin\_1\_modular\_double
  - fflas\_c.h, [954](#)
  - fflas\_lvl1.C, [973](#)
- fscalin\_2\_modular\_double
  - fflas\_c.h, [957](#)
  - fflas\_lvl2.C, [978](#)
- fspm

- FFLAS, [147](#)
- FFLAS::sparse\_details, [234–236](#)
- FFLAS::sparse\_details\_impl, [250](#), [257](#), [258](#), [263](#), [264](#), [268](#), [277](#)
- fspmm\_dispatch
  - FFLAS::sparse\_details, [233](#), [234](#)
- fspmm\_mone
  - FFLAS::sparse\_details\_impl, [251](#), [259](#), [269](#)
- fspmm\_mone\_simd\_aligned
  - FFLAS::sparse\_details\_impl, [252](#), [259](#), [270](#)
- fspmm\_mone\_simd\_unaligned
  - FFLAS::sparse\_details\_impl, [252](#), [260](#), [270](#)
- fspmm\_one
  - FFLAS::sparse\_details\_impl, [251](#), [258](#), [269](#)
- fspmm\_one\_simd\_aligned
  - FFLAS::sparse\_details\_impl, [252](#), [259](#), [270](#)
- fspmm\_one\_simd\_unaligned
  - FFLAS::sparse\_details\_impl, [252](#), [259](#), [270](#)
- fspmm\_simd\_aligned
  - FFLAS::sparse\_details\_impl, [251](#), [258](#)
- fspmm\_simd\_unaligned
  - FFLAS::sparse\_details\_impl, [251](#), [258](#)
- fspmv
  - FFLAS, [147](#), [154](#)
  - FFLAS::sparse\_details, [231–233](#), [241](#)
  - FFLAS::sparse\_details\_impl, [253](#), [260](#), [264](#), [271](#), [273](#), [274](#), [278](#), [280](#)
- fspmv\_dispatch
  - FFLAS::sparse\_details, [231](#)
- fspmv\_mone
  - FFLAS::sparse\_details\_impl, [253](#), [254](#), [261](#), [271](#), [272](#), [275](#), [281](#), [282](#)
- fspmv\_mone\_simd
  - FFLAS::sparse\_details\_impl, [275](#), [281](#)
- fspmv\_one
  - FFLAS::sparse\_details\_impl, [253](#), [254](#), [260](#), [261](#), [271](#), [272](#), [274](#), [275](#), [281](#)
- fspmv\_one\_simd
  - FFLAS::sparse\_details\_impl, [275](#), [281](#)
- fspmv\_simd
  - FFLAS::sparse\_details\_impl, [274](#), [280](#)
- fsquare
  - FFLAS, [92](#), [93](#), [181](#)
- fsquare\_3\_modular\_double
  - fflas\_c.h, [961](#)
  - fflas\_lvl3.C, [982](#)
- fsquareCommon
  - FFLAS::Protected, [220](#)
- fsub
  - FFLAS, [78](#), [81](#), [166](#), [174](#)
- fsub\_1\_modular\_double
  - fflas\_c.h, [955](#)
  - fflas\_lvl1.C, [974](#)
- fsub\_2\_modular\_double
  - fflas\_c.h, [958](#)
  - fflas\_lvl2.C, [979](#)
- fsubin
  - FFLAS, [78](#), [82](#), [174](#)
- fsubin\_1\_modular\_double
  - fflas\_c.h, [955](#)
  - fflas\_lvl1.C, [975](#)
- fsubin\_2\_modular\_double
  - fflas\_c.h, [959](#)
  - fflas\_lvl2.C, [979](#)
- fswap
  - FFLAS, [135](#), [165](#)
- fswap\_1\_modular\_double
  - fflas\_c.h, [955](#)
  - fflas\_lvl1.C, [974](#)
- fsyr2k
  - FFLAS, [118](#)
- fsyrk
  - FFLAS, [119–126](#)
- fsyrk.C, [772](#)
  - CUBE, [772](#)
  - GFOPS, [772](#)
  - main, [772](#)
  - TTimer, [772](#)
- fsyrk\_convert
  - FFLAS::Protected, [223](#)
- fsyrk\_strassen
  - FFLAS, [126](#), [144](#)
- fsytrf
  - FFPACK, [316](#), [317](#)
- fsytrf.C, [773](#)
  - CUBE, [773](#)
  - GFOPS, [773](#)
  - main, [773](#)
  - TTimer, [773](#)
- fsytrf\_BC\_Crout
  - FFPACK, [357](#)
- fsytrf\_BC\_RL
  - FFPACK, [357](#)
- fsytrf\_LOW\_RPM\_BC\_Crout
  - FFPACK, [358](#)
- fsytrf\_nonunit
  - FFPACK, [317](#), [358](#), [359](#)
- fsytrf\_RPM
  - FFPACK, [359](#)
- fsytrf\_UP\_RPM
  - FFPACK, [358](#)
- fsytrf\_UP\_RPM\_BC\_Crout
  - FFPACK, [358](#)
- fsytrf\_UP\_RPM\_BC\_RL
  - FFPACK, [357](#)
- ftmm
  - FFLAS, [127](#), [128](#), [178](#)
- ftmm\_3\_modular\_double
  - fflas\_c.h, [960](#)
  - fflas\_lvl3.C, [981](#)
- ftmmLeftLowerNoTransNonUnit< Element >, [464](#)
- ftmmLeftLowerNoTransUnit< Element >, [464](#)
- ftmmLeftLowerTransNonUnit< Element >, [465](#)
- ftmmLeftLowerTransUnit< Element >, [465](#)
- ftmmLeftUpperNoTransNonUnit< Element >, [465](#)
- ftmmLeftUpperNoTransUnit< Element >, [465](#)

- ftmmLeftUpperTransNonUnit< Element >, 465
- ftmmLeftUpperTransUnit< Element >, 465
- ftmmRightLowerNoTransNonUnit< Element >, 465
- ftmmRightLowerNoTransUnit< Element >, 465
- ftmmRightLowerTransNonUnit< Element >, 466
- ftmmRightLowerTransUnit< Element >, 466
- ftmmRightUpperNoTransNonUnit< Element >, 466
- ftmmRightUpperNoTransUnit< Element >, 466
- ftmmRightUpperTransNonUnit< Element >, 466
- ftmmRightUpperTransUnit< Element >, 466
- ftmv
  - FFLAS, 143
- ftsm
  - FFLAS, 128–130, 143, 147, 177
- ftsm\_3\_modular\_double
  - fflas\_c.h, 960
  - fflas\_lvl3.C, 981
- ftsmLeftLowerNoTransNonUnit< Element >, 466
- ftsmLeftLowerNoTransUnit< Element >, 466
- ftsmLeftLowerTransNonUnit< Element >, 467
- ftsmLeftLowerTransUnit< Element >, 467
- ftsmLeftUpperNoTransNonUnit< Element >, 467
- ftsmLeftUpperNoTransUnit< Element >, 467
- ftsmLeftUpperTransNonUnit< Element >, 467
- ftsmLeftUpperTransUnit< Element >, 467
- ftsmRightLowerNoTransNonUnit< Element >, 468
- ftsmRightLowerNoTransUnit< Element >, 468
- ftsmRightLowerTransNonUnit< Element >, 468
- ftsmRightLowerTransUnit< Element >, 468
- ftsmRightUpperNoTransNonUnit< Element >, 468
- ftsmRightUpperNoTransUnit< Element >, 468
- ftsmRightUpperTransNonUnit< Element >, 468
- ftsmRightUpperTransUnit< Element >, 468
- ftssyr2k
  - FFPACK, 316
- ftstr
  - FFPACK, 315
- ftsv
  - FFLAS, 130, 177
- ftsv\_2\_modular\_double
  - fflas\_c.h, 960
  - fflas\_lvl2.C, 980
- fttri
  - FFPACK, 314, 371
- fttri.C, 773
  - CUBE, 774
  - GFOPS, 774
  - main, 774
  - TTimer, 774
- fttri\_modular\_double
  - ffpack.C, 989
  - ffpack\_c.h, 1011
- fttrm
  - FFPACK, 315, 372
- fttrm\_modular\_double
  - ffpack.C, 990
  - ffpack\_c.h, 1011
- fzero
  - FFLAS, 133, 137, 161, 167
- fzero\_1\_modular\_double
  - fflas\_c.h, 953
  - fflas\_lvl1.C, 973
- fzero\_2\_modular\_double
  - fflas\_c.h, 956
  - fflas\_lvl2.C, 976
- gather
  - Simd128\_impl< true, true, false, 2 >, 565, 567
  - Simd128\_impl< true, true, false, 4 >, 575, 577
  - Simd128\_impl< true, true, false, 8 >, 585, 587
  - Simd128\_impl< true, true, true, 2 >, 595
  - Simd128\_impl< true, true, true, 4 >, 604
  - Simd128\_impl< true, true, true, 8 >, 613
  - Simd256\_impl< true, false, true, 8 >, 624
  - Simd256\_impl< true, true, false, 2 >, 632, 635
  - Simd256\_impl< true, true, false, 4 >, 644, 646, 649
  - Simd256\_impl< true, true, false, 8 >, 660, 663
  - Simd256\_impl< true, true, true, 2 >, 670
  - Simd256\_impl< true, true, true, 4 >, 681, 687
  - Simd256\_impl< true, true, true, 8 >, 697
  - Simd512\_impl< true, false, true, 8 >, 706
  - Simd512\_impl< true, true, false, 8 >, 713, 716
  - Simd512\_impl< true, true, true, 8 >, 724
- GaussJordan
  - FFPACK::Protected, 398
- GCC\_VERSION
  - fflas-ffpack-config.h, 823
- gebp
  - FFLAS::details, 210
- genData
  - benchmark-fgemv.C, 789
- GenericTag, 469
- genInputs
  - ScalFunctions< Element >, 552
- genInputsWithZero
  - ScalFunctions< Element >, 552
- get
  - Simd128\_impl< true, true, false, 8 >, 588
  - Simd128\_impl< true, true, true, 8 >, 613
  - Simd256\_impl< true, true, false, 8 >, 663
  - Simd256\_impl< true, true, true, 8 >, 697
- get\_default\_random\_generator
  - ScalFunctionsBase< Element, typename enable\_if< is\_floating\_point< Element >::value >::type >, 557
  - ScalFunctionsBase< Element, typename enable\_if< is\_integral< Element >::value >::type >, 559
- getArgumentValue
  - FFLAS, 188
- getDataType
  - FFLAS, 153, 154
- getEchelonForm
  - FFPACK, 342, 343
- getEchelonForm< FFLAS\_FIELD< FFLAS\_ELT > >
  - FFPACK, 380
- getEchelonForm\_modular\_double

- ffpack.C, [1001](#)
- ffpack\_c.h, [1021](#)
- getEchelonFormin\_modular\_double
  - ffpack.C, [1001](#)
  - ffpack\_c.h, [1021](#)
- getEchelonTransform
  - FFPACK, [344](#)
- getEchelonTransform< FFLAS\_FIELD< FFLAS\_ELT > >
  - FFPACK, [380](#)
- getEchelonTransform\_modular\_double
  - ffpack.C, [1002](#)
  - ffpack\_c.h, [1021](#)
- getListArgs
  - args-parser.h, [1039](#)
- getLTBruhatGen
  - FFPACK, [347](#), [348](#)
- getReducedEchelonForm
  - FFPACK, [344](#), [345](#)
- getReducedEchelonForm< FFLAS\_FIELD< FFLAS\_ELT > >
  - FFPACK, [381](#)
- getReducedEchelonForm\_modular\_double
  - ffpack.C, [1002](#)
  - ffpack\_c.h, [1022](#)
- getReducedEchelonFormin\_modular\_double
  - ffpack.C, [1002](#)
  - ffpack\_c.h, [1022](#)
- getReducedEchelonTransform
  - FFPACK, [346](#)
- getReducedEchelonTransform< FFLAS\_FIELD< FFLAS\_ELT > >
  - FFPACK, [381](#)
- getReducedEchelonTransform\_modular\_double
  - ffpack.C, [1002](#)
  - ffpack\_c.h, [1022](#)
- getSeed
  - FFLAS, [193](#)
- getStat
  - FFLAS, [156](#)
- getStatus
  - TestOneMethod< Simd >, [764](#)
- getTestName
  - TestOneMethod< Simd >, [764](#)
- getTLBSize
  - FFLAS, [193](#)
- getTriangular
  - FFPACK, [341](#), [342](#)
- getTriangular< FFLAS\_FIELD< FFLAS\_ELT > >
  - FFPACK, [379](#), [380](#)
- getTriangular\_modular\_double
  - ffpack.C, [1001](#)
  - ffpack\_c.h, [1020](#)
- getTriangularin\_modular\_double
  - ffpack.C, [1001](#)
  - ffpack\_c.h, [1020](#)
- getTridiagonal
  - FFPACK, [359](#)
- gf2ModularBalanced
  - regression-check.C, [1051](#)
- GFOPS
  - arithprog.C, [771](#)
  - charpoly.C, [808](#)
  - fsyrk.C, [772](#)
  - fsytrf.C, [773](#)
  - fttrtri.C, [774](#)
  - pluq.C, [810](#)
  - winograd.C, [775](#)
- Givaro, [403](#)
- GRAIN
  - benchmark-fgemm-rns.C, [785](#)
- Grain, [469](#)
- greater
  - ScalFunctions< Element >, [555](#)
  - Simd128\_impl< true, true, false, 2 >, [565](#)
  - Simd128\_impl< true, true, false, 4 >, [575](#)
  - Simd128\_impl< true, true, false, 8 >, [585](#)
  - Simd128\_impl< true, true, true, 2 >, [600](#)
  - Simd128\_impl< true, true, true, 4 >, [609](#)
  - Simd128\_impl< true, true, true, 8 >, [618](#)
  - Simd256\_impl< true, false, true, 8 >, [628](#)
  - Simd256\_impl< true, true, false, 2 >, [633](#)
  - Simd256\_impl< true, true, false, 4 >, [645](#), [647](#)
  - Simd256\_impl< true, true, false, 8 >, [661](#)
  - Simd256\_impl< true, true, true, 2 >, [675](#)
  - Simd256\_impl< true, true, true, 4 >, [686](#), [692](#)
  - Simd256\_impl< true, true, true, 8 >, [701](#)
  - Simd512\_impl< true, false, true, 8 >, [709](#)
  - Simd512\_impl< true, true, false, 8 >, [714](#)
  - Simd512\_impl< true, true, true, 8 >, [729](#)
- greater\_eq
  - ScalFunctions< Element >, [555](#)
  - Simd128\_impl< true, true, false, 2 >, [566](#)
  - Simd128\_impl< true, true, false, 4 >, [576](#)
  - Simd128\_impl< true, true, false, 8 >, [586](#)
  - Simd128\_impl< true, true, true, 2 >, [600](#)
  - Simd128\_impl< true, true, true, 4 >, [609](#)
  - Simd128\_impl< true, true, true, 8 >, [618](#)
  - Simd256\_impl< true, false, true, 8 >, [628](#)
  - Simd256\_impl< true, true, false, 2 >, [633](#)
  - Simd256\_impl< true, true, false, 4 >, [645](#), [647](#)
  - Simd256\_impl< true, true, false, 8 >, [661](#)
  - Simd256\_impl< true, true, true, 2 >, [675](#)
  - Simd256\_impl< true, true, true, 4 >, [686](#), [692](#)
  - Simd256\_impl< true, true, true, 8 >, [701](#)
  - Simd512\_impl< true, false, true, 8 >, [710](#)
  - Simd512\_impl< true, true, false, 8 >, [714](#)
  - Simd512\_impl< true, true, true, 8 >, [729](#)
- hadd
  - Simd256\_impl< true, false, true, 8 >, [628](#)
  - Simd512\_impl< true, false, true, 8 >, [710](#)
- hadd\_to\_scal
  - Simd128\_impl< true, true, false, 2 >, [567](#)
  - Simd128\_impl< true, true, false, 4 >, [577](#)
  - Simd128\_impl< true, true, false, 8 >, [587](#)
  - Simd128\_impl< true, true, true, 2 >, [600](#)

- Simd128\_impl< true, true, true, 4 >, [609](#)
- Simd128\_impl< true, true, true, 8 >, [618](#)
- Simd256\_impl< true, false, true, 8 >, [629](#)
- Simd256\_impl< true, true, false, 2 >, [634](#)
- Simd256\_impl< true, true, false, 4 >, [646](#), [648](#)
- Simd256\_impl< true, true, false, 8 >, [663](#)
- Simd256\_impl< true, true, true, 2 >, [676](#)
- Simd256\_impl< true, true, true, 4 >, [686](#), [692](#)
- Simd256\_impl< true, true, true, 8 >, [702](#)
- Simd512\_impl< true, false, true, 8 >, [710](#)
- Simd512\_impl< true, true, false, 8 >, [716](#)
- Simd512\_impl< true, true, true, 8 >, [729](#)
- half\_t
  - Simd256\_impl< true, true, false, 2 >, [632](#)
  - Simd256\_impl< true, true, false, 4 >, [643](#)
  - Simd256\_impl< true, true, false, 8 >, [660](#)
  - Simd256\_impl< true, true, true, 2 >, [669](#)
  - Simd256\_impl< true, true, true, 4 >, [680](#)
  - Simd256\_impl< true, true, true, 8 >, [696](#)
  - Simd512\_impl< true, true, false, 8 >, [713](#)
  - Simd512\_impl< true, true, true, 8 >, [723](#)
- has\_equal
  - FFLAS, [73](#)
- has\_minus
  - FFLAS, [73](#)
- has\_minus\_eq
  - FFLAS, [73](#)
- has\_minus\_eq\_impl< C >, [469](#)
  - value, [469](#)
- has\_minus\_impl< C >, [469](#)
  - value, [470](#)
- has\_mul
  - FFLAS, [73](#)
- has\_mul\_eq
  - FFLAS, [74](#)
- has\_mul\_eq\_impl< C >, [470](#)
  - value, [470](#)
- has\_mul\_impl< C >, [470](#)
  - value, [470](#)
- has\_operation< T >, [470](#)
  - value, [471](#)
- has\_plus
  - FFLAS, [73](#)
- has\_plus\_eq
  - FFLAS, [73](#)
- has\_plus\_eq\_impl< C >, [471](#)
  - value, [471](#)
- has\_plus\_impl< C >, [471](#)
  - value, [471](#)
- HAVE\_BIG\_ENDIAN
  - config.h, [801](#)
- HAVE\_BLAS
  - config.h, [801](#)
- HAVE\_CBLAS
  - config.h, [802](#)
- HAVE\_CXX11
  - config.h, [802](#)
- HAVE\_DLFCN\_H
  - config.h, [802](#)
- HAVE\_FLOAT\_H
  - config.h, [802](#)
- HAVE\_INT128
  - config.h, [802](#)
- HAVE\_INTPTR\_T\_H
  - config.h, [802](#)
- HAVE\_LAPACK
  - config.h, [802](#)
- HAVE\_LIMITS\_H
  - config.h, [802](#)
- HAVE\_PTHREAD\_H
  - config.h, [802](#)
- HAVE\_STDDEF\_H
  - config.h, [802](#)
- HAVE\_STDINT\_H
  - config.h, [802](#)
- HAVE\_STDIO\_H
  - config.h, [802](#)
- HAVE\_STDLIB\_H
  - config.h, [802](#)
- HAVE\_STRING\_H
  - config.h, [802](#)
- HAVE\_STRINGS\_H
  - config.h, [802](#)
- HAVE\_SYS\_STAT\_H
  - config.h, [803](#)
- HAVE\_SYS\_TIME\_H
  - config.h, [803](#)
- HAVE\_SYS\_TYPES\_H
  - config.h, [803](#)
- HAVE\_UNISTD\_H
  - config.h, [803](#)
- HelperFlag, [471](#)
  - aut, [472](#)
  - coo, [472](#)
  - csr, [472](#)
  - ell, [472](#)
  - none, [472](#)
  - pm1, [472](#)
- HelperMod
  - HelperMod< Field, ElementCategories::MachineIntTag >, [472](#)
  - HelperMod< Field, FFLAS::ElementCategories::ArbitraryPrecIntTag >, [473](#)
  - HelperMod< Field, FFLAS::ElementCategories::FixedPrecIntTag >, [474](#)
  - HelperMod< Field, FFLAS::ElementCategories::MachineFloatTag >, [475](#)
  - HelperMod< Field, ElementCategories::MachineIntTag >, [472](#)
  - HelperMod, [472](#)
  - invp, [473](#)
  - max, [473](#)
  - min, [473](#)
  - p, [473](#)
  - pow50rem, [473](#)
- HelperMod< Field, ElementTraits >, [472](#)



HelperMod< Field, FFLAS::ElementCategories::ArbitraryPrecisionTag, 859  
     >, 473  
     HelperMod, 473  
     p, 474  
 HelperMod< Field, FFLAS::ElementCategories::FixedPrecisionTag, 859  
     >, 474  
     HelperMod, 474  
     p, 474  
 HelperMod< Field, FFLAS::ElementCategories::MachineFloatTag, 859  
     >, 474  
     HelperMod, 475  
     invp, 475  
     max, 475  
     min, 475  
     p, 475  
 helpString  
     Argument, 407  
 HYB\_ZO  
     FFLAS, 76  
 hyb\_zo.h, 904  
 hyb\_zo\_pspmm.inl, 904  
     \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_pspmm\_INL, 905  
 hyb\_zo\_pspmv.inl, 905  
     \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_pspmv\_INL, 905  
 hyb\_zo\_spm.inl, 905  
     \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_spm\_INL, 906  
 hyb\_zo\_spmv.inl, 906  
     \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_spmv\_INL, 906  
 hyb\_zo\_utils.inl, 906  
     \_\_FFLASFFPACK\_fflas\_sparse\_HYB\_ZO\_utils\_INL, 906  
 Hybrid, 475  
 Hybrid\_KGF\_LUK\_MinPoly  
     FFPACK::Protected, 401  
 idamax\_  
     config-blas.h, 819  
 igebb11  
     FFLAS::details, 209  
 igebb14  
     FFLAS::details, 208  
 igebb21  
     FFLAS::details, 208  
 igebb24  
     FFLAS::details, 207  
 igebb41  
     FFLAS::details, 208  
 igebb44  
     FFLAS::details, 207  
 igebp  
     FFLAS::details, 209  
 igemm  
     FFLAS::Protected, 226  
 igemm.doxy, 858  
 igemm.h, 858  
 \_\_FFLASFFPACK\_fflas\_igemm\_igemm\_INL, 859  
 igemm\_  
     FFLAS, 131  
 \_\_FFLASFFPACK\_fflas\_igemm\_igemm\_colmajor  
     FFLAS::Protected, 226  
 igemm\_kernels.h, 859  
 igemm\_kernels.inl, 860  
 \_\_FFLASFFPACK\_fflas\_igemm\_igemm\_kernels\_INL, 861  
 igemm\_tools.h, 861  
 igemm\_tools.inl, 861  
     \_\_FFLASFFPACK\_fflas\_igemm\_igemm\_tools\_INL, 861  
 in\_range  
     flimits.h, 1046  
 index\_t  
     fflas\_sparse.h, 884  
     parallel.h, 1031  
 InfNorm  
     FFLAS, 77  
     begin, 475, 477  
     Info, 476, 477  
     operator=, 476, 477  
     perm, 476, 477  
     size, 476, 477  
 BlockTransposeSIMD< Field, Simd, >, 413  
 init  
     FieldSimd< \_Field >, 452  
     rns\_double, 526, 527  
     rns\_double\_extended, 537, 538  
     RNSInteger< RNS >, 542  
     RNSIntegerMod< RNS >, 546, 547  
 init\_transpose  
     rns\_double, 527  
 init\_y  
     FFLAS::sparse\_details, 231  
 initA  
     MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, 498  
 initB  
     MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, 498  
 initC  
     MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, 498  
 initOut  
     MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, 498  
 INLINE  
     fflas\_simd.h, 871  
 inplace  
     Bench< Elt >, 413  
 inputs  
     TestOneMethod< Simd >, 765  
 INST\_OR\_DECL

- fflas\_L1\_inst.C, [961](#)
- fflas\_L1\_inst.h, [962](#)
- fflas\_L2\_inst.C, [965](#)
- fflas\_L2\_inst.h, [966](#)
- fflas\_L3\_inst.C, [968](#)
- fflas\_L3\_inst.h, [970](#)
- ffpack\_inst.C, [1023](#)
- ffpack\_inst.h, [1024](#)
- integer
  - rns\_double, [525](#)
  - rns\_double\_extended, [536](#)
  - RNSInteger< RNS >, [540](#)
  - RNSIntegerMod< RNS >, [545](#)
- Interfaces, [42](#)
- interfaces.doxy, [949](#)
- IntType
  - ScalFunctionsBase< Element, typename enable\_if< is\_floating\_point< Element >::value >::type >::FloatingPointTestDistribution, [464](#)
- inv
  - RNSIntegerMod< RNS >, [548](#)
- Invert
  - FFPACK, [325](#), [326](#), [374](#)
- Invert2
  - FFPACK, [326](#), [374](#)
- Invert2\_modular\_double
  - ffpack.C, [997](#)
  - ffpack\_c.h, [1016](#)
- Invert\_modular\_double
  - ffpack.C, [996](#)
  - ffpack\_c.h, [1016](#)
- Invertin\_modular\_double
  - ffpack.C, [996](#)
  - ffpack\_c.h, [1016](#)
- invp
  - HelperMod< Field, ElementCategories::MachineIntTag >, [473](#)
  - HelperMod< Field, FFLAS::ElementCategories::MachineIntTag >, [475](#)
- is\_all\_same< Args >, [478](#)
- is\_all\_same< T, Args... >, [478](#)
  - value, [478](#)
- is\_all\_same<>, [478](#)
  - value, [478](#)
- is\_same\_element
  - Bench< Elt >, [411](#)
  - NoSimd< T >, [518](#)
  - Simd128\_impl< true, true, false, 2 >, [564](#)
  - Simd128\_impl< true, true, false, 4 >, [574](#)
  - Simd128\_impl< true, true, false, 8 >, [584](#)
  - Simd128\_impl< true, true, true, 2 >, [595](#)
  - Simd128\_impl< true, true, true, 4 >, [604](#)
  - Simd128\_impl< true, true, true, 8 >, [613](#)
  - Simd256\_impl< true, false, true, 8 >, [623](#)
  - Simd256\_impl< true, true, false, 2 >, [631](#)
  - Simd256\_impl< true, true, false, 4 >, [643](#)
  - Simd256\_impl< true, true, false, 8 >, [660](#)
  - Simd256\_impl< true, true, true, 2 >, [670](#)
  - Simd256\_impl< true, true, true, 4 >, [680](#), [681](#)
  - Simd256\_impl< true, true, true, 8 >, [696](#)
  - Simd512\_impl< true, false, true, 8 >, [705](#)
  - Simd512\_impl< true, true, false, 8 >, [713](#)
  - Simd512\_impl< true, true, true, 8 >, [724](#)
  - Test< Elt >, [760](#)
- is\_simd< T >, [478](#)
  - type, [479](#)
  - value, [479](#)
- isMOne
  - RNSInteger< RNS >, [541](#)
  - RNSIntegerMod< RNS >, [545](#)
- isOdd
  - FFPACK, [382](#)
- isOne
  - RNSInteger< RNS >, [541](#)
  - RNSIntegerMod< RNS >, [545](#)
- IsSingular
  - FFPACK, [331](#), [376](#)
- IsSingular\_modular\_double
  - ffpack.C, [997](#)
  - ffpack\_c.h, [1017](#)
- isSparseMatrix< Field, M >, [479](#)
- isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::COO > >, [479](#)
- isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::COO\_ZO > >, [479](#)
- isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::CSR > >, [480](#)
- isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::CSR\_HYB > >, [480](#)
- isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::CSR\_ZO > >, [480](#)
- isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::ELL > >, [481](#)
- isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::ELL\_simd > >, [481](#)
- isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::ELL\_simd\_ZO > >, [481](#)
- isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::ELL\_ZO > >, [482](#)
- isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::HYB\_ZO > >, [482](#)
- isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::SELL > >, [482](#)
- isSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::SELL\_ZO > >, [482](#)
- isSparseMatrixMKLFormat< F, M >, [483](#)
- isSparseMatrixSimdFormat< F, M >, [483](#)
- isZero
  - RNSInteger< RNS >, [541](#)
  - RNSIntegerMod< RNS >, [545](#)
- isZOSparseMatrix< F, M >, [483](#)
- isZOSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::COO\_ZO > >, [484](#)
- isZOSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::CSR\_ZO > >, [484](#)
- isZOSparseMatrix< Field, Sparse< Field, SparseMa-



- trix\_t::ELL\_simd\_ZO > >, [484](#)
- isZOSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::ELL\_ZO > >, [484](#)
- isZOSparseMatrix< Field, Sparse< Field, SparseMatrix\_t::SELL\_ZO > >, [485](#)
- Iterative, [485](#)
- iters
  - Bench< Elt >, [413](#)
- kaapi\_routines.inl, [1030](#)
- \_\_FFLASFFPACK\_KAAPI\_ROUTINES\_INL, [1030](#)
- KellerGehrig
  - FFPACK::Protected, [398](#)
- KGFast
  - FFPACK::Protected, [398](#)
- KGFast\_generalized
  - FFPACK::Protected, [399](#)
- kmax
  - Sparse< \_Field, SparseMatrix\_t::COO >, [734](#)
  - Sparse< \_Field, SparseMatrix\_t::COO\_ZO >, [736](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR >, [737](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR\_HYB >, [739](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR\_ZO >, [741](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL >, [742](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd >, [744](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd\_ZO >, [746](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_ZO >, [747](#)
  - Sparse< \_Field, SparseMatrix\_t::HYB\_ZO >, [749](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL >, [751](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, [753](#)
- KrylovElim
  - FFPACK, [376](#)
- KrylovElim\_modular\_double
  - ffpack.C, [997](#)
  - ffpack\_c.h, [1016](#)
- lapack.C, [1050](#)
- \_\_FFLASFFPACK\_CONFIGURATION, [1050](#)
- \_\_FFLASFFPACK\_HAVE\_LAPACK, [1050](#)
- main, [1050](#)
- LAPACKPerm2MathPerm
  - FFPACK, [308](#)
  - ffpack.C, [986](#)
  - ffpack\_c.h, [1008](#)
- launch\_fger
  - test-fger.C, [1066](#)
- launch\_fger\_dispatch
  - test-fger.C, [1066](#)
- launch\_MM
  - test-fgemm.C, [1062](#)
- launch\_MM\_dispatch
  - test-fgemm-check.C, [1061](#)
  - test-fgemm.C, [1063](#)
- launch\_MV
  - test-fgemv.C, [1064](#)
- launch\_MV\_dispatch
  - test-fgemv.C, [1065](#)
- launch\_test
  - test-charpoly.C, [1053](#)
  - test-lu.C, [1088](#)
  - test-quasisep.C, [1095](#)
- launch\_wino
  - benchmark-wino.C, [800](#)
- LazyTag, [485](#)
- LD
  - fflas\_transpose.h, [913](#)
- ld
  - Sparse< \_Field, SparseMatrix\_t::ELL >, [743](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd >, [744](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd\_ZO >, [746](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_ZO >, [748](#)
- LeadingSubmatrixRankProfiles
  - FFPACK, [338](#)
  - ffpack.C, [999](#)
  - ffpack\_c.h, [1019](#)
- lesser
  - ScalFunctions< Element >, [555](#)
  - Simd128\_impl< true, true, false, 2 >, [565](#)
  - Simd128\_impl< true, true, false, 4 >, [576](#)
  - Simd128\_impl< true, true, false, 8 >, [586](#)
  - Simd128\_impl< true, true, true, 2 >, [600](#)
  - Simd128\_impl< true, true, true, 4 >, [609](#)
  - Simd128\_impl< true, true, true, 8 >, [618](#)
  - Simd256\_impl< true, false, true, 8 >, [627](#)
  - Simd256\_impl< true, true, false, 2 >, [633](#)
  - Simd256\_impl< true, true, false, 4 >, [645](#), [647](#)
  - Simd256\_impl< true, true, false, 8 >, [661](#)
  - Simd256\_impl< true, true, true, 2 >, [675](#)
  - Simd256\_impl< true, true, true, 4 >, [686](#), [692](#)
  - Simd256\_impl< true, true, true, 8 >, [701](#)
  - Simd512\_impl< true, false, true, 8 >, [709](#)
  - Simd512\_impl< true, true, false, 8 >, [714](#)
  - Simd512\_impl< true, true, true, 8 >, [729](#)
- lesser\_eq
  - ScalFunctions< Element >, [555](#)
  - Simd128\_impl< true, true, false, 2 >, [566](#)
  - Simd128\_impl< true, true, false, 4 >, [576](#)
  - Simd128\_impl< true, true, false, 8 >, [586](#)
  - Simd128\_impl< true, true, true, 2 >, [600](#)
  - Simd128\_impl< true, true, true, 4 >, [609](#)
  - Simd128\_impl< true, true, true, 8 >, [618](#)
  - Simd256\_impl< true, false, true, 8 >, [627](#)
  - Simd256\_impl< true, true, false, 2 >, [633](#)
  - Simd256\_impl< true, true, false, 4 >, [645](#), [647](#)
  - Simd256\_impl< true, true, false, 8 >, [661](#)
  - Simd256\_impl< true, true, true, 2 >, [676](#)
  - Simd256\_impl< true, true, true, 4 >, [686](#), [692](#)
  - Simd256\_impl< true, true, true, 8 >, [701](#)
  - Simd512\_impl< true, false, true, 8 >, [709](#)
  - Simd512\_impl< true, true, false, 8 >, [715](#)
  - Simd512\_impl< true, true, true, 8 >, [729](#)
- limits< char >, [485](#)
- digits, [486](#)
- max, [486](#)

- min, [486](#)
- T, [486](#)
- limits< double >, [486](#)
  - digits, [487](#)
  - max, [486](#)
  - min, [486](#)
  - T, [486](#)
- limits< float >, [487](#)
  - digits, [487](#)
  - max, [487](#)
  - min, [487](#)
  - T, [487](#)
- limits< Givaro::Integer >, [487](#)
  - max, [488](#)
  - min, [488](#)
  - T, [488](#)
- limits< int >, [488](#)
  - digits, [488](#)
  - max, [488](#)
  - min, [488](#)
  - T, [488](#)
- limits< long >, [488](#)
  - digits, [489](#)
  - max, [489](#)
  - min, [489](#)
  - T, [489](#)
- limits< long long >, [489](#)
  - digits, [490](#)
  - max, [489](#)
  - min, [489](#)
  - T, [489](#)
- limits< Reclnt::rint< K > >, [490](#)
  - max, [490](#)
  - min, [490](#)
  - T, [490](#)
- limits< Reclnt::ruint< K > >, [490](#)
  - max, [491](#)
  - min, [491](#)
  - T, [491](#)
- limits< short int >, [491](#)
  - digits, [491](#)
  - max, [491](#)
  - min, [491](#)
  - T, [491](#)
- limits< signed char >, [492](#)
  - digits, [492](#)
  - max, [492](#)
  - min, [492](#)
  - T, [492](#)
- limits< T >, [485](#)
- limits< unsigned char >, [492](#)
  - digits, [493](#)
  - max, [493](#)
  - min, [493](#)
  - T, [492](#)
- limits< unsigned int >, [493](#)
  - digits, [493](#)
  - max, [493](#)
- min, [493](#)
- T, [493](#)
- limits< unsigned long >, [493](#)
  - digits, [494](#)
  - max, [494](#)
  - min, [494](#)
  - T, [494](#)
- limits< unsigned long long >, [494](#)
  - digits, [494](#)
  - max, [494](#)
  - min, [494](#)
  - T, [494](#)
- limits< unsigned short int >, [495](#)
  - digits, [495](#)
  - max, [495](#)
  - min, [495](#)
  - T, [495](#)
- load
  - Simd128\_impl< true, true, false, 2 >, [565](#), [567](#)
  - Simd128\_impl< true, true, false, 4 >, [575](#), [577](#)
  - Simd128\_impl< true, true, false, 8 >, [585](#), [588](#)
  - Simd128\_impl< true, true, true, 2 >, [595](#)
  - Simd128\_impl< true, true, true, 4 >, [604](#)
  - Simd128\_impl< true, true, true, 8 >, [613](#)
  - Simd256\_impl< true, false, true, 8 >, [624](#)
  - Simd256\_impl< true, true, false, 2 >, [632](#), [635](#)
  - Simd256\_impl< true, true, false, 4 >, [644](#), [646](#), [650](#)
  - Simd256\_impl< true, true, false, 8 >, [661](#), [663](#)
  - Simd256\_impl< true, true, true, 2 >, [671](#)
  - Simd256\_impl< true, true, true, 4 >, [681](#), [687](#)
  - Simd256\_impl< true, true, true, 8 >, [697](#)
  - Simd512\_impl< true, false, true, 8 >, [706](#)
  - Simd512\_impl< true, true, false, 8 >, [714](#), [717](#)
  - Simd512\_impl< true, true, true, 8 >, [725](#)
- loadu
  - Simd128\_impl< true, true, false, 2 >, [565](#), [567](#)
  - Simd128\_impl< true, true, false, 4 >, [575](#), [578](#)
  - Simd128\_impl< true, true, false, 8 >, [585](#), [588](#)
  - Simd128\_impl< true, true, true, 2 >, [595](#)
  - Simd128\_impl< true, true, true, 4 >, [604](#)
  - Simd128\_impl< true, true, true, 8 >, [614](#)
  - Simd256\_impl< true, false, true, 8 >, [624](#)
  - Simd256\_impl< true, true, false, 2 >, [632](#), [635](#)
  - Simd256\_impl< true, true, false, 4 >, [644](#), [647](#), [650](#)
  - Simd256\_impl< true, true, false, 8 >, [661](#), [663](#)
  - Simd256\_impl< true, true, true, 2 >, [671](#)
  - Simd256\_impl< true, true, true, 4 >, [681](#), [688](#)
  - Simd256\_impl< true, true, true, 8 >, [697](#)
  - Simd512\_impl< true, false, true, 8 >, [706](#)
  - Simd512\_impl< true, true, false, 8 >, [714](#), [717](#)
  - Simd512\_impl< true, true, true, 8 >, [725](#)
- LQUPtoInverseOfFullRankMinor
  - FFPACK, [352](#), [382](#)
- LT\_OBJDIR
  - config.h, [803](#)
- LTBruhatGen

- FFPACK, [347](#)
- LTQSorter
  - FFPACK, [349](#)
- LUdivine
  - FFPACK, [320](#), [360](#), [372](#)
- LUdivine\_construct
  - FFPACK::Protected, [397](#), [403](#)
- LUdivine\_gauss
  - FFPACK, [359](#), [373](#)
- LUdivine\_gauss\_modular\_double
  - ffpack\_c.h, [1012](#)
- LUdivine\_modular\_double
  - ffpack.C, [990](#)
  - ffpack\_c.h, [1012](#)
- LUdivine\_small
  - FFPACK, [360](#), [372](#)
- LUdivine\_small\_modular\_double
  - ffpack\_c.h, [1012](#)
- LUKrylov
  - FFPACK::Protected, [399](#)
- LUKrylov\_KGFast
  - FFPACK::Protected, [400](#)
- m
  - Bench< Elt >, [412](#)
  - Sparse< \_Field, SparseMatrix\_t::COO >, [734](#)
  - Sparse< \_Field, SparseMatrix\_t::COO\_ZO >, [736](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR >, [737](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR\_HYB >, [739](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR\_ZO >, [741](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL >, [743](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd >, [744](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd\_ZO >, [746](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_ZO >, [747](#)
  - Sparse< \_Field, SparseMatrix\_t::HYB\_ZO >, [749](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL >, [751](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, [753](#)
- MachineFloatTag, [495](#)
- MachineIntTag, [495](#)
- main
  - 101-fgemm.C, [1102](#)
  - 2x2-fgemm.C, [1103](#)
  - 2x2-ftrsv.C, [1103](#)
  - 2x2-pluq.C, [1103](#)
  - arithprog.C, [772](#)
  - benchmark-charpoly-mp.C, [776](#)
  - benchmark-charpoly.C, [777](#)
  - benchmark-checkers.C, [778](#)
  - benchmark-dgemm.C, [778](#)
  - benchmark-dgetrf.C, [779](#)
  - benchmark-dgetri.C, [780](#)
  - benchmark-dsytrf.C, [781](#)
  - benchmark-dtrsm.C, [781](#)
  - benchmark-dtrtri.C, [782](#)
  - benchmark-fadd-lvl2.C, [782](#)
  - benchmark-fdot.C, [783](#)
  - benchmark-fgemm-mp.C, [784](#)
  - benchmark-fgemm-rns.C, [786](#)
  - benchmark-fgemm.C, [786](#)
  - benchmark-fgemv-mp.C, [787](#)
  - benchmark-fgemv.C, [791](#)
  - benchmark-fgesv.C, [791](#)
  - benchmark-fsyr2k.C, [792](#)
  - benchmark-fsyrk.C, [792](#)
  - benchmark-fsytrf.C, [793](#)
  - benchmark-ftrsm-mp.C, [794](#)
  - benchmark-ftrsm.C, [794](#)
  - benchmark-ftrsv.C, [795](#)
  - benchmark-ftrtri.C, [795](#)
  - benchmark-inverse.C, [796](#)
  - benchmark-lqup-mp.C, [796](#)
  - benchmark-lqup.C, [797](#)
  - benchmark-pluq.C, [798](#)
  - benchmark-quasisep.C, [799](#)
  - benchmark-storage-transpose.C, [800](#)
  - benchmark-wino.C, [800](#)
  - cblas.C, [1048](#)
  - charpoly.C, [808](#), [809](#)
  - clapack.C, [1049](#)
  - cuda.C, [1049](#)
  - det.C, [809](#)
  - fblas.C, [1050](#)
  - fflas-101\_1.C, [1104](#)
  - fflas-101\_3.C, [1104](#)
  - fflas\_101.C, [1105](#)
  - fflas\_101\_lvl1.C, [1105](#)
  - ffpack-fgesv.C, [1105](#)
  - ffpack-solve.C, [1106](#)
  - fsyrk.C, [772](#)
  - fsytrf.C, [773](#)
  - ftrtri.C, [774](#)
  - lapack.C, [1050](#)
  - matmul.C, [809](#)
  - pluq.C, [810](#), [811](#)
  - rank.C, [811](#)
  - regression-check.C, [1051](#)
  - solve.C, [811](#)
  - test-charpoly-check.C, [1052](#)
  - test-charpoly.C, [1053](#)
  - test-compressQ.C, [1054](#)
  - test-det-check.C, [1055](#)
  - test-det.C, [1055](#)
  - test-echelon.C, [1057](#)
  - test-fadd.C, [1059](#)
  - test-fdot.C, [1060](#)
  - test-fgemm-check.C, [1061](#)
  - test-fgemm.C, [1063](#)
  - test-fgemv.C, [1065](#)
  - test-fger.C, [1067](#)
  - test-fgesv.C, [1068](#)
  - test-finit.C, [1069](#)
  - test-fscal.C, [1071](#)
  - test-fsyr2k.C, [1072](#)
  - test-fsyrk.C, [1074](#)
  - test-fsytrf.C, [1075](#)

- test-ftrmm.C, [1076](#)
- test-ftrmv.C, [1077](#)
- test-ftrsm-check.C, [1078](#)
- test-ftrsm.C, [1079](#)
- test-ftrssyr2k.C, [1080](#)
- test-ftrstr.C, [1081](#)
- test-ftrsv.C, [1083](#)
- test-ftrtri.C, [1084](#)
- test-interfaces-c.c, [1084](#)
- test-invert-check.C, [1085](#)
- test-io.C, [1085](#)
- test-lu.C, [1089](#)
- test-maxdelayeddim.C, [1090](#)
- test-minpoly.C, [1091](#)
- test-multifile2.C, [1091](#)
- test-nullspace.C, [1093](#)
- test-permutations.C, [1093](#)
- test-pluq-check.C, [1094](#)
- test-quasisep.C, [1096](#)
- test-rankprofiles.C, [1097](#)
- test-rpm.C, [1097](#)
- test-simd.C, [1100](#)
- test-solve.C, [1101](#)
- test-storage-transpose.C, [1102](#)
- winograd.C, [775](#)
- mainpage.doxy, [808](#)
- mask\_high
  - Simd128\_impl< true, true, false, 8 >, [591](#)
  - Simd128\_impl< true, true, true, 8 >, [618](#)
  - Simd256\_impl< true, true, false, 8 >, [667](#)
  - Simd256\_impl< true, true, true, 8 >, [702](#)
  - Simd512\_impl< true, true, false, 8 >, [720](#)
  - Simd512\_impl< true, true, true, 8 >, [730](#)
- mask\_t
  - read\_sparse.h, [907](#)
- maskstore
  - Simd512\_impl< true, true, false, 8 >, [714](#), [717](#)
  - Simd512\_impl< true, true, true, 8 >, [725](#)
- MatF2MatD\_Triangular
  - FFLAS::Protected, [227](#)
- MatF2MatFI\_Triangular
  - FFLAS::Protected, [227](#)
- MathPerm2LAPACKPerm
  - FFPACK, [308](#)
  - ffpack.C, [986](#)
  - ffpack\_c.h, [1008](#)
- Matio.h, [1046](#)
  - read\_field, [1047](#)
  - write\_field, [1047](#)
- matmul.C, [809](#)
  - main, [809](#)
- matmul.doxy, [838](#)
- Matrix Multiplication Algorithms, [42](#)
- MatrixApplyS
  - FFPACK, [362](#), [363](#)
- MatrixApplyS\_modular\_double
  - ffpack.C, [986](#)
  - ffpack\_c.h, [1008](#)
- MatrixApplyT
  - FFPACK, [364](#)
- MatrixApplyT\_modular\_double
  - ffpack.C, [987](#)
  - ffpack\_c.h, [1008](#)
- MatVecMinPoly
  - FFPACK, [330](#), [375](#)
  - FFPACK::Protected, [400](#)
- max
  - HelperMod< Field, ElementCategories::MachineIntTag >, [473](#)
  - HelperMod< Field, FFLAS::ElementCategories::MachineFloatTag >, [475](#)
  - limits< char >, [486](#)
  - limits< double >, [486](#)
  - limits< float >, [487](#)
  - limits< Givaro::Integer >, [488](#)
  - limits< int >, [488](#)
  - limits< long >, [489](#)
  - limits< long long >, [489](#)
  - limits< Reclnt::rint< K > >, [490](#)
  - limits< Reclnt::ruint< K > >, [491](#)
  - limits< short int >, [491](#)
  - limits< signed char >, [492](#)
  - limits< unsigned char >, [493](#)
  - limits< unsigned int >, [493](#)
  - limits< unsigned long >, [494](#)
  - limits< unsigned long long >, [494](#)
  - limits< unsigned short int >, [495](#)
- max3
  - FFLAS, [77](#)
- max4
  - FFLAS, [77](#)
- MAX\_THREADS
  - parallel.h, [1032](#)
- MAX\_WITH\_SIZE\_T
  - test-maxdelayeddim.C, [1090](#)
- maxCardinality
  - FFLAS, [156](#)
- maxCardinality< Givaro::Modular< int32\_t > >
  - FFLAS, [156](#)
- maxCardinality< Givaro::Modular< int64\_t > >
  - FFLAS, [156](#)
- maxCol
  - StatsMatrix, [756](#)
- maxColDifference
  - StatsMatrix, [756](#)
- MaxDelayedDim
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [498](#)
- maxElement
  - RNSIntegerMod< RNS >, [546](#)
- maxpy
  - FieldSimd< \_Field >, [455](#)
- maxpyin
  - FieldSimd< \_Field >, [456](#)
- maxRow
  - StatsMatrix, [755](#)

- maxrow
  - Sparse< \_Field, SparseMatrix\_t::COO >, [735](#)
  - Sparse< \_Field, SparseMatrix\_t::COO\_ZO >, [736](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR >, [738](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR\_HYB >, [739](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR\_ZO >, [741](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL >, [743](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd >, [744](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd\_ZO >, [746](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_ZO >, [748](#)
  - Sparse< \_Field, SparseMatrix\_t::HYB\_ZO >, [749](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL >, [751](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, [753](#)
- maxRowDifference
  - StatsMatrix, [756](#)
- MaxStorableValue
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait >, [500](#)
- MG\_DEFAULT
  - benchmark-fgemm-mp.C, [784](#)
  - benchmark-fgemv-mp.C, [787](#)
- min
  - HelperMod< Field, ElementCategories::MachineIntTag >, [473](#)
  - HelperMod< Field, FFLAS::ElementCategories::MachineFloatTag >, [475](#)
  - limits< char >, [486](#)
  - limits< double >, [486](#)
  - limits< float >, [487](#)
  - limits< Givaro::Integer >, [488](#)
  - limits< int >, [488](#)
  - limits< long >, [489](#)
  - limits< long long >, [489](#)
  - limits< RecInt::rint< K > >, [490](#)
  - limits< RecInt::ruint< K > >, [491](#)
  - limits< short int >, [491](#)
  - limits< signed char >, [492](#)
  - limits< unsigned char >, [493](#)
  - limits< unsigned int >, [493](#)
  - limits< unsigned long >, [494](#)
  - limits< unsigned long long >, [494](#)
  - limits< unsigned short int >, [495](#)
- min3
  - FFLAS, [77](#)
- min4
  - FFLAS, [77](#)
- min\_types
  - FFLAS::Protected, [224](#), [225](#)
- minCardinality
  - FFLAS, [156](#)
- minCol
  - StatsMatrix, [756](#)
- minColDifference
  - StatsMatrix, [756](#)
- minElement
  - RNSIntegerMod< RNS >, [546](#)
- MinPoly
  - FFPACK, [329](#), [375](#)
- minRow
  - StatsMatrix, [756](#)
- minRowDifference
  - StatsMatrix, [756](#)
- MKL\_CONFIG, [403](#)
- MKLSparseMatrixFormat
  - FFLAS, [73](#)
- MMHelper
  - MMHelper< FFPACK::RNSInteger< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, [501](#), [502](#)
  - MMHelper< FFPACK::RNSIntegerMod< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, [503](#), [504](#)
  - MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< Dest >, ParSeqTrait >, [505](#), [506](#)
  - MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeqTrait >, [507](#)
  - MMHelper< Field, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, [509](#)
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait >, [497](#)
  - MMHelper< FFPACK::RNSInteger< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, [501](#)
  - MMHelper, [501](#), [502](#)
  - normA, [502](#)
  - normB, [502](#)
  - operator<<, [502](#)
  - parseq, [502](#)
  - recLevel, [502](#)
  - Self\_t, [501](#)
  - setNorm, [502](#)
  - MMHelper< FFPACK::RNSIntegerMod< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, [503](#)
  - MMHelper, [503](#), [504](#)
  - normA, [504](#)
  - normB, [504](#)
  - operator<<, [504](#)
  - parseq, [504](#)
  - recLevel, [504](#)
  - Self\_t, [503](#)
  - setNorm, [504](#)
  - MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< Dest >, ParSeqTrait >, [505](#)
  - MMHelper, [505](#), [506](#)
  - operator<<, [506](#)
  - parseq, [506](#)
  - recLevel, [506](#)
  - Self\_t, [505](#)
  - MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeqTrait >, [506](#)
  - MMHelper, [507](#)

- normA, [508](#)
- normB, [508](#)
- operator<<, [508](#)
- parseq, [508](#)
- recLevel, [508](#)
- Self\_t, [507](#)
- setNorm, [508](#)
- MMHelper< Field, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, [508](#)
  - MMHelper, [509](#)
  - operator<<, [509](#)
  - parseq, [510](#)
  - recLevel, [510](#)
  - Self\_t, [509](#)
- MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait >, [496](#)
  - Amax, [500](#)
  - Amin, [500](#)
  - Aunfit, [498](#)
  - Bmax, [500](#)
  - Bmin, [500](#)
  - Bunfit, [498](#)
  - checkA, [498](#)
  - checkB, [499](#)
  - checkOut, [499](#)
  - Cmax, [500](#)
  - Cmin, [500](#)
  - DelayedField, [497](#)
  - delayedField, [500](#)
  - DelayedField\_t, [497](#)
  - DFElt, [497](#)
  - FieldMax, [500](#)
  - FieldMin, [499](#)
  - initA, [498](#)
  - initB, [498](#)
  - initC, [498](#)
  - initOut, [498](#)
  - MaxDelayedDim, [498](#)
  - MaxStorableValue, [500](#)
  - MMHelper, [497](#)
  - operator<<, [499](#)
  - Outmax, [500](#)
  - Outmin, [500](#)
  - parseq, [500](#)
  - recLevel, [499](#)
  - Self\_t, [497](#)
  - setOutBounds, [498](#)
- mod
  - FieldSimd< \_Field >, [454](#)
  - Simd128\_impl< true, true, false, 2 >, [571](#)
  - Simd128\_impl< true, true, false, 4 >, [581](#)
  - Simd128\_impl< true, true, false, 8 >, [591](#)
  - Simd128\_impl< true, true, true, 2 >, [600](#)
  - Simd128\_impl< true, true, true, 4 >, [609](#)
  - Simd128\_impl< true, true, true, 8 >, [618](#)
  - Simd256\_impl< true, false, true, 8 >, [629](#)
  - Simd256\_impl< true, true, false, 2 >, [639](#)
  - Simd256\_impl< true, true, false, 4 >, [656](#), [657](#)
  - Simd256\_impl< true, true, false, 8 >, [667](#)
  - Simd256\_impl< true, true, true, 2 >, [676](#)
  - Simd256\_impl< true, true, true, 4 >, [686](#), [693](#)
  - Simd256\_impl< true, true, true, 8 >, [702](#)
  - Simd512\_impl< true, true, false, 8 >, [720](#)
  - Simd512\_impl< true, true, true, 8 >, [730](#)
- MODE
  - parallel.h, [1034](#)
- ModeTraits< Field >, [510](#)
  - value, [510](#)
- ModeTraits< Givaro::Modular< Element, Compute > >, [510](#)
  - value, [510](#)
- ModeTraits< Givaro::Modular< Givaro::Integer, Compute > >, [511](#)
  - value, [511](#)
- ModeTraits< Givaro::Modular< int16\_t, Compute > >, [511](#)
  - value, [511](#)
- ModeTraits< Givaro::Modular< int32\_t, Compute > >, [511](#)
  - value, [511](#)
- ModeTraits< Givaro::Modular< int64\_t, uint64\_t > >, [512](#)
  - value, [512](#)
- ModeTraits< Givaro::Modular< int8\_t, Compute > >, [512](#)
  - value, [512](#)
- ModeTraits< Givaro::Modular< RecInt::ruint< K >, Compute > >, [512](#)
  - value, [512](#)
- ModeTraits< Givaro::Modular< uint16\_t, Compute > >, [513](#)
  - value, [513](#)
- ModeTraits< Givaro::Modular< uint32\_t, Compute > >, [513](#)
  - value, [513](#)
- ModeTraits< Givaro::Modular< uint8\_t, Compute > >, [513](#)
  - value, [513](#)
- ModeTraits< Givaro::ModularBalanced< Element > >, [514](#)
  - value, [514](#)
- ModeTraits< Givaro::ModularBalanced< Givaro::Integer > >, [514](#)
  - value, [514](#)
- ModeTraits< Givaro::ModularBalanced< int16\_t > >, [514](#)
  - value, [514](#)
- ModeTraits< Givaro::ModularBalanced< int32\_t > >, [515](#)
  - value, [515](#)
- ModeTraits< Givaro::ModularBalanced< int8\_t > >, [515](#)
  - value, [515](#)
- ModeTraits< Givaro::Montgomery< T > >, [515](#)
  - value, [515](#)
- ModeTraits< Givaro::ZRing< double > >, [515](#)



- value, [516](#)
  - ModeTraits< Givaro::ZRing< float > >, [516](#)
    - value, [516](#)
  - ModeTraits< Givaro::ZRing< Givaro::Integer > >, [516](#)
    - value, [516](#)
  - ModField
    - rns\_double, [525](#)
    - rns\_double\_extended, [536](#)
    - RNSIntegerMod< RNS >, [545](#)
  - modp
    - FFLAS::vectorised, [287](#)
    - FFLAS::vectorised::unswitch, [289](#), [290](#)
  - ModularBalanced< T >, [516](#)
  - ModularTag, [516](#)
  - mOne
    - RNSInteger< RNS >, [543](#)
    - RNSIntegerMod< RNS >, [549](#)
  - mone
    - FFLAS, [76](#)
    - Sparse< \_Field, SparseMatrix\_t::HYB\_ZO >, [749](#)
  - MonotonicApplyP
    - FFPACK, [310](#)
  - MonotonicCompress
    - FFPACK, [361](#)
  - MonotonicCompressCycles
    - FFPACK, [361](#)
  - MonotonicCompressMorePivots
    - FFPACK, [361](#)
  - MonotonicExpand
    - FFPACK, [362](#)
  - Montgomery< T >, [517](#)
  - mul
    - FieldSimd< \_Field >, [454](#)
    - RNSIntegerMod< RNS >, [547](#)
    - ScalFunctions< Element >, [553](#)
    - Simd128\_impl< true, true, false, 2 >, [570](#)
    - Simd128\_impl< true, true, false, 4 >, [580](#)
    - Simd128\_impl< true, true, false, 8 >, [590](#)
    - Simd128\_impl< true, true, true, 2 >, [598](#)
    - Simd128\_impl< true, true, true, 4 >, [607](#)
    - Simd128\_impl< true, true, true, 8 >, [616](#)
    - Simd256\_impl< true, false, true, 8 >, [626](#)
    - Simd256\_impl< true, true, false, 2 >, [638](#)
    - Simd256\_impl< true, true, false, 4 >, [654](#)
    - Simd256\_impl< true, true, false, 8 >, [666](#)
    - Simd256\_impl< true, true, true, 2 >, [673](#)
    - Simd256\_impl< true, true, true, 4 >, [684](#), [690](#)
    - Simd256\_impl< true, true, true, 8 >, [699](#)
    - Simd512\_impl< true, false, true, 8 >, [708](#)
    - Simd512\_impl< true, true, false, 8 >, [719](#)
    - Simd512\_impl< true, true, true, 8 >, [727](#)
  - mul\_r
    - FieldSimd< \_Field >, [454](#)
  - mulhi
    - ScalFunctionsBase< Element, typename enable\_if< is\_integral< Element >::value >::type >, [559](#)
    - Simd128\_impl< true, true, false, 2 >, [566](#)
    - Simd128\_impl< true, true, false, 4 >, [576](#)
  - Simd128\_impl< true, true, false, 8 >, [586](#)
  - Simd128\_impl< true, true, true, 2 >, [598](#)
  - Simd128\_impl< true, true, true, 4 >, [607](#)
  - Simd128\_impl< true, true, true, 8 >, [616](#)
  - Simd256\_impl< true, true, false, 2 >, [633](#)
  - Simd256\_impl< true, true, false, 4 >, [645](#), [648](#)
  - Simd256\_impl< true, true, false, 8 >, [662](#)
  - Simd256\_impl< true, true, true, 2 >, [674](#)
  - Simd256\_impl< true, true, true, 4 >, [684](#), [690](#)
  - Simd256\_impl< true, true, true, 8 >, [700](#)
  - Simd512\_impl< true, true, false, 8 >, [715](#)
  - Simd512\_impl< true, true, true, 8 >, [727](#)
- Simd128\_impl< true, true, false, 8 >, [586](#)
- Simd128\_impl< true, true, true, 2 >, [598](#)
- Simd128\_impl< true, true, true, 4 >, [607](#)
- Simd128\_impl< true, true, true, 8 >, [616](#)
- Simd256\_impl< true, true, false, 2 >, [633](#)
- Simd256\_impl< true, true, false, 4 >, [645](#), [647](#)
- Simd256\_impl< true, true, false, 8 >, [662](#)
- Simd256\_impl< true, true, true, 2 >, [674](#)
- Simd256\_impl< true, true, true, 4 >, [684](#), [690](#)
- Simd256\_impl< true, true, true, 8 >, [699](#)
- Simd512\_impl< true, true, false, 8 >, [715](#)
- Simd512\_impl< true, true, true, 8 >, [727](#)
- mulhi\_fast
  - Simd128\_impl< true, true, false, 8 >, [591](#)
  - Simd128\_impl< true, true, true, 8 >, [618](#)
  - Simd256\_impl< true, true, false, 8 >, [667](#)
  - Simd256\_impl< true, true, true, 8 >, [702](#)
  - Simd512\_impl< true, true, false, 8 >, [720](#)
  - Simd512\_impl< true, true, true, 8 >, [730](#)
- mulin
  - FieldSimd< \_Field >, [454](#)
  - ScalFunctions< Element >, [554](#)
  - Simd256\_impl< true, false, true, 8 >, [626](#)
  - Simd512\_impl< true, false, true, 8 >, [708](#)
- mullo
  - ScalFunctionsBase< Element, typename enable\_if< is\_integral< Element >::value >::type >, [559](#)
  - Simd128\_impl< true, true, false, 2 >, [570](#)
  - Simd128\_impl< true, true, false, 4 >, [580](#)
  - Simd128\_impl< true, true, false, 8 >, [586](#)
  - Simd128\_impl< true, true, true, 2 >, [598](#)
  - Simd128\_impl< true, true, true, 4 >, [607](#)
  - Simd128\_impl< true, true, true, 8 >, [616](#)
  - Simd256\_impl< true, true, false, 2 >, [638](#)
  - Simd256\_impl< true, true, false, 4 >, [654](#)
  - Simd256\_impl< true, true, false, 8 >, [662](#)
  - Simd256\_impl< true, true, true, 2 >, [673](#)
  - Simd256\_impl< true, true, true, 4 >, [684](#), [690](#)
  - Simd256\_impl< true, true, true, 8 >, [699](#)
  - Simd512\_impl< true, true, false, 8 >, [715](#)
  - Simd512\_impl< true, true, true, 8 >, [727](#)
- mulx
  - ScalFunctionsBase< Element, typename enable\_if< is\_integral< Element >::value >::type >, [559](#)
  - Simd128\_impl< true, true, false, 2 >, [566](#)
  - Simd128\_impl< true, true, false, 4 >, [576](#)
  - Simd128\_impl< true, true, false, 8 >, [586](#)
  - Simd128\_impl< true, true, true, 2 >, [598](#)
  - Simd128\_impl< true, true, true, 4 >, [607](#)
  - Simd128\_impl< true, true, true, 8 >, [616](#)
  - Simd256\_impl< true, true, false, 2 >, [633](#)
  - Simd256\_impl< true, true, false, 4 >, [645](#), [648](#)
  - Simd256\_impl< true, true, false, 8 >, [662](#)
  - Simd256\_impl< true, true, true, 2 >, [674](#)
  - Simd256\_impl< true, true, true, 4 >, [684](#), [690](#)
  - Simd256\_impl< true, true, true, 8 >, [700](#)
  - Simd512\_impl< true, true, false, 8 >, [715](#)
  - Simd512\_impl< true, true, true, 8 >, [727](#)

- mvcnt
  - test-lu.C, [1089](#)
- n
  - Bench< Elt >, [412](#)
  - count\_nonconst\_lvalue\_reference< const T &, O... >, [439](#)
  - count\_nonconst\_lvalue\_reference< T &, O... >, [440](#)
  - count\_nonconst\_lvalue\_reference< T, O... >, [440](#)
  - count\_nonconst\_lvalue\_reference<>, [440](#)
  - Sparse< \_Field, SparseMatrix\_t::COO >, [734](#)
  - Sparse< \_Field, SparseMatrix\_t::COO\_ZO >, [736](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR >, [737](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR\_HYB >, [739](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR\_ZO >, [741](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL >, [743](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd >, [744](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd\_ZO >, [746](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_ZO >, [747](#)
  - Sparse< \_Field, SparseMatrix\_t::HYB\_ZO >, [749](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL >, [751](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, [753](#)
- name
  - TestOneMethod< Simd >, [765](#)
- nb\_lref
  - TestOneMethod< Simd >, [764](#)
- nChunks
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd >, [745](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd\_ZO >, [746](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL >, [751](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, [753](#)
- nDenseCols
  - StatsMatrix, [757](#)
- nDenseRows
  - StatsMatrix, [757](#)
- need\_field\_characteristic< Field >, [517](#)
- value, [517](#)
- need\_field\_characteristic< Givaro::Modular< Field > >, [517](#)
- value, [517](#)
- need\_field\_characteristic< Givaro::ModularBalanced< Field > >, [517](#)
- value, [517](#)
- NeedDoublePreAddReduction
  - FFLAS::Protected, [219](#)
- NeedPreAddReduction
  - FFLAS::Protected, [218](#)
- NeedPreAxyReduction
  - FFLAS::Protected, [224](#)
- NeedPreScalReduction
  - FFLAS::Protected, [223](#), [224](#)
- NeedPreSubReduction
  - FFLAS::Protected, [218](#)
- neg
  - RNSIntegerMod< RNS >, [547](#)
- nElements
  - Sparse< \_Field, SparseMatrix\_t::COO >, [735](#)
  - Sparse< \_Field, SparseMatrix\_t::COO\_ZO >, [736](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR >, [738](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR\_HYB >, [739](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR\_ZO >, [741](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL >, [743](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd >, [744](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd\_ZO >, [746](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_ZO >, [748](#)
  - Sparse< \_Field, SparseMatrix\_t::HYB\_ZO >, [749](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL >, [751](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, [753](#)
- nEmptyCols
  - StatsMatrix, [757](#)
- nEmptyColsEnd
  - StatsMatrix, [757](#)
- nEmptyRows
  - StatsMatrix, [757](#)
- newD
  - FFPACK::Protected, [401](#)
- NEWWINO
  - fgemm\_winograd.inl, [838](#)
- nMOnes
  - Sparse< \_Field, SparseMatrix\_t::CSR\_HYB >, [740](#)
  - StatsMatrix, [755](#)
- nnz
  - Sparse< \_Field, SparseMatrix\_t::COO >, [734](#)
  - Sparse< \_Field, SparseMatrix\_t::COO\_ZO >, [736](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR >, [737](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR\_HYB >, [739](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR\_ZO >, [741](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL >, [743](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd >, [744](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_simd\_ZO >, [746](#)
  - Sparse< \_Field, SparseMatrix\_t::ELL\_ZO >, [748](#)
  - Sparse< \_Field, SparseMatrix\_t::HYB\_ZO >, [749](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL >, [751](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, [753](#)
  - StatsMatrix, [755](#)
- none
  - HelperFlag, [472](#)
- nOnes
  - Sparse< \_Field, SparseMatrix\_t::CSR\_HYB >, [740](#)
  - StatsMatrix, [755](#)
- nonsquare\_inplace\_v1
  - FFLAS::ftranspose\_impl, [194](#)
- nonsquare\_inplace\_v2
  - FFLAS::ftranspose\_impl, [194](#)
- NonZeroRandomMatrix
  - FFPACK, [382](#), [383](#)
- normA



- MMHelper< FFPACK::RNSInteger< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, 502
- MMHelper< FFPACK::RNSIntegerMod< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, 504
- MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeqTrait >, 508
- normB
  - MMHelper< FFPACK::RNSInteger< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, 502
  - MMHelper< FFPACK::RNSIntegerMod< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, 504
  - MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeqTrait >, 508
- NORML\_MOD
  - fflas\_simd.h, 871
- NoSimd< T >, 518
  - aligned\_allocator, 518
  - aligned\_vector, 518
  - alignment, 519
  - compliant, 519
  - is\_same\_element, 518
  - scalar\_t, 518
  - type\_string, 519
  - valid, 519
  - vect\_size, 519
  - vect\_t, 518
- NoSimdSparseMatrix
  - FFLAS, 72
- NOSPLIT
  - parallel.h, 1036
- not\_inplace
  - FFLAS::\_ftranspose\_impl, 194
- nOthers
  - Sparse< \_Field, SparseMatrix\_t::CSR\_HYB >, 740
  - StatsMatrix, 755
- NotMKLSparseMatrixFormat
  - FFLAS, 73
- NotZOSparseMatrix
  - FFLAS, 72
- NullSpaceBasis
  - FFPACK, 334, 378
- NullSpaceBasis\_modular\_double
  - ffpack.C, 999
  - ffpack\_c.h, 1018
- NUM\_THREADS
  - parallel.h, 1032
- NUMARGS
  - parallel.h, 1034
- number\_kind
  - FFLAS, 76
- numthreads
  - Parallel< C, P >, 520
  - Sequential, 561
- one
  - FFLAS, 76
  - RNSInteger< RNS >, 543
  - RNSIntegerMod< RNS >, 549
  - Sparse< \_Field, SparseMatrix\_t::HYB\_ZO >, 749
- OPENBLAS\_NUM\_THREADS
  - config.h, 803
- operator!=
  - rns\_double\_elt\_cstptr, 532
  - rns\_double\_elt\_ptr, 535
- operator<
  - rns\_double\_elt\_cstptr, 532
  - rns\_double\_elt\_ptr, 535
- operator<<
  - Compose< H1, H2 >, 428
  - FFLAS, 153
- MMHelper< FFPACK::RNSInteger< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, 502
- MMHelper< FFPACK::RNSIntegerMod< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, 504
- MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< Dest >, ParSeqTrait >, 506
- MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeqTrait >, 508
- MMHelper< Field, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, 509
- MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait >, 499
- Parallel< C, P >, 520
- Sequential, 561
- test-fsytrf.C, 1074
- test-simd.C, 1100
- operator()
  - callLUdivine\_small< double >, 415
  - callLUdivine\_small< Element >, 415
  - callLUdivine\_small< float >, 416
  - Failure, 448
  - readMyMachineType< Field, mpz\_t >, 524
  - readMyMachineType< Field, T >, 523
  - RNSInteger< RNS >::RandIter, 521
  - RNSIntegerMod< RNS >::RandIter, 522
  - rnsRandIter< RNS >, 550, 551
  - ScalFunctionsBase< Element, typename enable\_if< is\_floating\_point< Element >::value >::type >::FloatingPointTestDistribution, 464
  - tfn\_minus, 765
  - tfn\_minus\_eq, 766
  - tfn\_mul, 766
  - tfn\_mul\_eq, 766
  - tfn\_plus, 767
  - tfn\_plus\_eq, 767
- operator+
  - rns\_double\_elt\_cstptr, 532

- rns\_double\_elt\_ptr, [534](#)
- operator++
  - rns\_double\_elt\_cstptr, [532](#)
  - rns\_double\_elt\_ptr, [534](#)
- operator+=
  - rns\_double\_elt\_cstptr, [532](#)
  - rns\_double\_elt\_ptr, [535](#)
- operator-
  - rns\_double\_elt\_cstptr, [532](#)
  - rns\_double\_elt\_ptr, [535](#)
- operator--
  - rns\_double\_elt\_cstptr, [532](#)
  - rns\_double\_elt\_ptr, [534](#)
- operator-=
  - rns\_double\_elt\_cstptr, [532](#)
  - rns\_double\_elt\_ptr, [535](#)
- operator=
  - Coo< Field >, [436](#)
  - Coo< ValT, IdxT >, [435](#), [438](#)
  - FieldSimd< \_Field >, [452](#)
  - Info, [476](#), [477](#)
  - rns\_double\_elt\_cstptr, [532](#)
  - rns\_double\_elt\_ptr, [535](#)
- operator&
  - rns\_double\_elt, [530](#)
  - rns\_double\_elt\_cstptr, [531](#), [532](#)
  - rns\_double\_elt\_ptr, [534](#), [535](#)
- operator[]
  - rns\_double\_elt\_cstptr, [531](#), [532](#)
  - rns\_double\_elt\_ptr, [534](#)
- operator\*
  - rns\_double\_elt\_cstptr, [531](#)
  - rns\_double\_elt\_ptr, [534](#)
- other
  - FFLAS, [76](#)
  - rns\_double\_elt\_cstptr, [533](#)
  - rns\_double\_elt\_ptr, [535](#)
- Outmax
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [500](#)
- Outmin
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-Trait >, [500](#)
- outputs\_scalar
  - TestOneMethod< Simd >, [765](#)
- outputs\_simd
  - TestOneMethod< Simd >, [765](#)
- p
  - HelperMod< Field, ElementCategories::MachineIntTag >, [473](#)
  - HelperMod< Field, FFLAS::ElementCategories::ArbitraryPrecisionTag >, [474](#)
  - HelperMod< Field, FFLAS::ElementCategories::FixedPrecisionTag >, [474](#)
  - HelperMod< Field, FFLAS::ElementCategories::MachineFloatTag >, [475](#)
- pack
  - ScalFunctions< Element >, [556](#)
- Simd128\_impl< true, true, false, 2 >, [569](#)
  - Simd128\_impl< true, true, false, 4 >, [579](#)
  - Simd128\_impl< true, true, false, 8 >, [589](#)
  - Simd128\_impl< true, true, true, 2 >, [597](#)
  - Simd128\_impl< true, true, true, 4 >, [606](#)
  - Simd128\_impl< true, true, true, 8 >, [615](#)
  - Simd256\_impl< true, false, true, 8 >, [625](#)
  - Simd256\_impl< true, true, false, 2 >, [636](#)
  - Simd256\_impl< true, true, false, 4 >, [652](#)
  - Simd256\_impl< true, true, false, 8 >, [665](#)
  - Simd256\_impl< true, true, true, 2 >, [672](#)
  - Simd256\_impl< true, true, true, 4 >, [683](#), [689](#)
  - Simd256\_impl< true, true, true, 8 >, [698](#)
  - Simd512\_impl< true, false, true, 8 >, [707](#)
  - Simd512\_impl< true, true, false, 8 >, [718](#)
  - Simd512\_impl< true, true, true, 8 >, [726](#)
- pack\_even
  - ScalFunctions< Element >, [556](#)
  - Simd128\_impl< true, true, false, 2 >, [569](#)
  - Simd128\_impl< true, true, false, 4 >, [579](#)
  - Simd128\_impl< true, true, false, 8 >, [589](#)
  - Simd128\_impl< true, true, true, 2 >, [597](#)
  - Simd128\_impl< true, true, true, 4 >, [606](#)
  - Simd128\_impl< true, true, true, 8 >, [615](#)
  - Simd256\_impl< true, false, true, 8 >, [625](#)
  - Simd256\_impl< true, true, false, 2 >, [636](#)
  - Simd256\_impl< true, true, false, 4 >, [652](#)
  - Simd256\_impl< true, true, false, 8 >, [665](#)
  - Simd256\_impl< true, true, true, 2 >, [672](#)
  - Simd256\_impl< true, true, true, 4 >, [683](#), [689](#)
  - Simd256\_impl< true, true, true, 8 >, [698](#)
  - Simd512\_impl< true, false, true, 8 >, [707](#)
  - Simd512\_impl< true, true, false, 8 >, [718](#)
  - Simd512\_impl< true, true, true, 8 >, [726](#)
- pack\_lhs
  - FFLAS::details, [209](#)
- pack\_odd
  - ScalFunctions< Element >, [556](#)
  - Simd128\_impl< true, true, false, 2 >, [569](#)
  - Simd128\_impl< true, true, false, 4 >, [579](#)
  - Simd128\_impl< true, true, false, 8 >, [589](#)
  - Simd128\_impl< true, true, true, 2 >, [597](#)
  - Simd128\_impl< true, true, true, 4 >, [606](#)
  - Simd128\_impl< true, true, true, 8 >, [615](#)
  - Simd256\_impl< true, false, true, 8 >, [625](#)
  - Simd256\_impl< true, true, false, 2 >, [636](#)
  - Simd256\_impl< true, true, false, 4 >, [652](#)
  - Simd256\_impl< true, true, false, 8 >, [665](#)
  - Simd256\_impl< true, true, true, 2 >, [672](#)
  - Simd256\_impl< true, true, true, 4 >, [683](#), [689](#)
  - Simd256\_impl< true, true, true, 8 >, [698](#)
  - Simd512\_impl< true, false, true, 8 >, [707](#)
  - Simd512\_impl< true, true, false, 8 >, [718](#)
  - Simd512\_impl< true, true, true, 8 >, [726](#)
- pack\_rhs
  - FFLAS::details, [209](#)
- PACKAGE
  - config.h, [803](#)

PACKAGE\_BUGREPORT  
     config.h, [803](#)  
 PACKAGE\_NAME  
     config.h, [803](#)  
 PACKAGE\_STRING  
     config.h, [803](#)  
 PACKAGE\_TARNAME  
     config.h, [803](#)  
 PACKAGE\_URL  
     config.h, [803](#)  
 PACKAGE\_VERSION  
     config.h, [803](#)  
 PAR\_BLOCK  
     parallel.h, [1031](#)  
 Parallel  
     Parallel< C, P >, [520](#)  
 Parallel< C, P >, [519](#)  
     Cut, [519](#)  
     numthreads, [520](#)  
     operator<<, [520](#)  
     Parallel, [520](#)  
     Param, [519](#)  
     set\_numthreads, [520](#)  
 parallel.h, [1030](#)  
     \_\_FFLASFFPACK\_SEQUENTIAL, [1031](#)  
     BARRIER, [1031](#)  
     BEGIN\_PARALLEL\_MAIN, [1032](#)  
     CHECK\_DEPENDENCIES, [1031](#)  
     COMMA, [1034](#)  
     CONSTREFERENCE, [1032](#)  
     END\_PARALLEL\_MAIN, [1032](#)  
     FOR1D, [1033](#)  
     FOR2D, [1033](#)  
     FORBLOCK1D, [1032](#)  
     FORBLOCK2D, [1033](#)  
     index\_t, [1031](#)  
     MAX\_THREADS, [1032](#)  
     MODE, [1034](#)  
     NOSPLIT, [1036](#)  
     NUM\_THREADS, [1032](#)  
     NUMARGS, [1034](#)  
     PAR\_BLOCK, [1031](#)  
     PARFOR1D, [1033](#)  
     PARFOR2D, [1034](#)  
     PARFORBLOCK1D, [1033](#)  
     PARFORBLOCK2D, [1034](#)  
     PP\_ARG\_N, [1034](#)  
     PP\_NARG\_, [1034](#)  
     PP\_RSEQ\_N, [1036](#)  
     READ, [1032](#)  
     READWRITE, [1032](#)  
     RETURNPARAM, [1034](#)  
     SET\_THREADS, [1032](#)  
     splitt, [1036](#)  
     SPLITTER, [1036](#)  
     splitting\_0, [1036](#)  
     splitting\_1, [1036](#)  
     splitting\_2, [1036](#)  
     splitting\_3, [1036](#)  
     SYNCH\_GROUP, [1032](#)  
     TASK, [1031](#)  
     THREAD\_INDEX, [1032](#)  
     VALUE, [1032](#)  
     WAIT, [1031](#)  
     WRITE, [1032](#)  
 Param  
     Parallel< C, P >, [519](#)  
 PARFOR1D  
     parallel.h, [1033](#)  
 PARFOR2D  
     parallel.h, [1034](#)  
 PARFORBLOCK1D  
     parallel.h, [1033](#)  
 PARFORBLOCK2D  
     parallel.h, [1034](#)  
 parseArguments  
     FFLAS, [188](#)  
 parseq  
     MMHelper< FFPACK::RNSInteger< E >, Algo-  
         Trait, ModeCategories::DefaultTag, ParSeq-  
         Trait >, [502](#)  
     MMHelper< FFPACK::RNSIntegerMod< E >, Al-  
         goTrait, ModeCategories::DefaultTag, ParSeq-  
         Trait >, [504](#)  
     MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo<  
         Dest >, ParSeqTrait >, [506](#)  
     MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo<  
         ElementCategories::RNSElementTag >, >,  
         ParSeqTrait >, [508](#)  
     MMHelper< Field, AlgoTrait, ModeCategories::DefaultTag,  
         ParSeqTrait >, [510](#)  
     MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-  
         Trait >, [500](#)  
     TRSMHelper< ReclterTrait, ParSeqTrait >, [769](#)  
 PBASECASE\_K  
     ffpack\_ppluq.inl, [941](#)  
 pColumnEchelonForm  
     FFPACK, [321](#)  
 pColumnEchelonForm\_modular\_double  
     ffpack.C, [993](#)  
 pColumnEchelonForm\_modular\_float  
     ffpack.C, [994](#)  
 pColumnEchelonForm\_modular\_int32\_t  
     ffpack.C, [995](#)  
 pColumnRankProfile  
     FFPACK, [337](#)  
 pDet  
     FFPACK, [332](#)  
 perm  
     Info, [476](#), [477](#)  
     Sparse< \_Field, SparseMatrix\_t::SELL >, [751](#)  
     Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, [753](#)  
 PermApplyS  
     FFPACK, [363](#)  
 PermApplyS\_double  
     ffpack.C, [987](#)

- ffpack\_c.h, 1008
- PermApplyT
  - FFPACK, 365
- PermApplyT\_double
  - ffpack.C, 987
  - ffpack\_c.h, 1009
- pfadd
  - FFLAS, 79
- pfaddin
  - FFLAS, 79
- pfgemm
  - FFLAS, 144, 184–186
- pfgemm\_1D\_rec
  - FFLAS, 145
- pfgemm\_2D\_rec
  - FFLAS, 145
- pfgemm\_3D\_rec
  - FFLAS, 145
- pfgemm\_3D\_rec2
  - FFLAS, 146
- pfgemm\_variants.inl, 1037
- pfgemv.inl, 1037
- pftrand
  - FFLAS, 184
- pfreduce
  - FFLAS, 110
- pfspmm
  - FFLAS::sparse\_details, 237–239
  - FFLAS::sparse\_details\_impl, 254, 261, 262, 264–266, 276
- pfspmm\_dispatch
  - FFLAS::sparse\_details, 236, 237
- pfspmm\_mone
  - FFLAS::sparse\_details\_impl, 255
- pfspmm\_one
  - FFLAS::sparse\_details\_impl, 255
- pfspmm\_zo
  - FFLAS::sparse\_details\_impl, 266
- pfspmv
  - FFLAS::sparse\_details, 239, 240
  - FFLAS::sparse\_details\_impl, 256, 263, 266, 267, 272, 276–279
- pfspmv\_mone
  - FFLAS::sparse\_details\_impl, 257, 267, 268, 273, 279
- pfspmv\_one
  - FFLAS::sparse\_details\_impl, 256, 257, 267, 273, 279
- pfspmv\_task
  - FFLAS::sparse\_details\_impl, 256
- pfsub
  - FFLAS, 79
- pfsubin
  - FFLAS, 80
- pfzero
  - FFLAS, 184
- PLUQ
  - FFPACK, 318, 319, 368, 369, 372
- pluq.C, 810
  - CUBE, 810
  - GFOPS, 810
  - main, 810, 811
  - TTimer, 810
- PLUQ\_basecaseCROUT
  - FFPACK, 367
- PLUQ\_basecaseV2
  - FFPACK, 367
- PLUQ\_basecaseV3
  - FFPACK, 367
- PLUQ\_modular\_double
  - ffpack.C, 990
  - ffpack\_c.h, 1012
- PLUQtoEchelonPermutation
  - FFPACK, 347
  - ffpack.C, 1003
  - ffpack\_c.h, 1022
- pm1
  - HelperFlag, 472
- pMMH
  - TRSMHelper< ReclterTrait, ParSeqTrait >, 769
- pow50rem
  - HelperMod< Field, ElementCategories::MachineIntTag >, 473
- PP\_ARG\_N
  - parallel.h, 1034
- PP\_NARG\_
  - parallel.h, 1034
- PP\_RSEQ\_N
  - parallel.h, 1036
- pPLUQ
  - FFPACK, 319
- pRank
  - FFPACK, 331
- preamble
  - FFLAS, 190
- precompute\_cst
  - rns\_double, 526
  - rns\_double\_extended, 537
- pReducedColumnEchelonForm
  - FFPACK, 324
- pReducedColumnEchelonForm\_modular\_double
  - ffpack.C, 994
- pReducedColumnEchelonForm\_modular\_float
  - ffpack.C, 995
- pReducedColumnEchelonForm\_modular\_int32\_t
  - ffpack.C, 996
- pReducedRowEchelonForm
  - FFPACK, 325
- pReducedRowEchelonForm\_modular\_double
  - ffpack.C, 994
- pReducedRowEchelonForm\_modular\_float
  - ffpack.C, 995
- pReducedRowEchelonForm\_modular\_int32\_t
  - ffpack.C, 996
- prefetch
  - FFLAS, 192

- print
  - Failure, [449](#)
- printHelpMessage
  - args-parser.h, [1039](#)
- printPolynomial
  - test-charpoly-check.C, [1052](#)
- printvect
  - test-compressQ.C, [1054](#)
- productBruhatxTS
  - FFPACK, [351](#), [354](#)
- pRowEchelonForm
  - FFPACK, [322](#)
- pRowEchelonForm\_modular\_double
  - ffpack.C, [994](#)
- pRowEchelonForm\_modular\_float
  - ffpack.C, [995](#)
- pRowEchelonForm\_modular\_int32\_t
  - ffpack.C, [995](#)
- pRowRankProfile
  - FFPACK, [336](#)
- PSeq
  - benchmark-fgemm-rns.C, [786](#)
- pSolve
  - FFPACK, [334](#)
- PTRSM\_HYBRID\_THRESHOLD
  - fflas\_ptrsm.inl, [870](#)
- PURE
  - fflas\_simd.h, [871](#)
- queryCacheSizes
  - FFLAS, [193](#)
- queryL1CacheSize
  - FFLAS, [193](#)
- queryTopLevelCacheSize
  - FFLAS, [193](#)
- RandInt
  - FFPACK, [386](#)
- RandIter
  - RNSInteger< RNS >::RandIter, [521](#)
  - RNSIntegerMod< RNS >::RandIter, [522](#)
- random
  - RNSInteger< RNS >::RandIter, [521](#)
  - RNSIntegerMod< RNS >::RandIter, [522](#)
  - rnsRandIter< RNS >, [550](#), [551](#)
- RandomIndexSubset
  - FFPACK, [388](#)
- RandomKrylovPrecond
  - FFPACK::Protected, [400](#)
- RandomLTQSMMatrixWithRankandQSOrder
  - FFPACK, [395](#)
- RandomLTQSRankProfileMatrix
  - FFPACK, [389](#)
- RandomMatrix
  - FFPACK, [384](#)
- RandomMatrixWithDet
  - FFPACK, [394](#)
- RandomMatrixWithRank
  - FFPACK, [386](#), [387](#)
- RandomMatrixWithRankandRandomRPM
  - FFPACK, [392](#)
- RandomMatrixWithRankandRPM
  - FFPACK, [389](#), [390](#)
- RandomNullSpaceVector
  - FFPACK, [334](#), [352](#), [378](#)
- RandomNullSpaceVector\_modular\_double
  - ffpack.C, [998](#)
  - ffpack\_c.h, [1018](#)
- RandomPermutation
  - FFPACK, [388](#)
- RandomRankProfileMatrix
  - FFPACK, [388](#)
- RandomSymmetricMatrix
  - FFPACK, [386](#)
- RandomSymmetricMatrixWithRankandRandomRPM
  - FFPACK, [393](#)
- RandomSymmetricMatrixWithRankandRPM
  - FFPACK, [390](#), [391](#)
- RandomSymmetricRankProfileMatrix
  - FFPACK, [389](#)
- RandomTriangularMatrix
  - FFPACK, [385](#)
- Rank
  - FFPACK, [330](#), [331](#), [376](#)
- rank.C, [811](#)
  - main, [811](#)
- Rank\_modular\_double
  - ffpack.C, [997](#)
  - ffpack\_c.h, [1017](#)
- RankProfileFromLU
  - FFPACK, [337](#)
  - ffpack.C, [999](#)
  - ffpack\_c.h, [1019](#)
- READ
  - parallel.h, [1032](#)
- read\_field
  - Matio.h, [1047](#)
- read\_sparse.h, [907](#)
  - DNS\_BIN\_VER, [907](#)
  - mask\_t, [907](#)
- readDnsFormat
  - FFLAS, [154](#)
- readMachineType
  - FFLAS, [154](#)
- ReadMatrix
  - FFLAS, [190](#)
- readMyMachineType< Field, mpz\_t >, [523](#)
  - Element, [523](#)
  - Element\_ptr, [523](#)
  - operator(), [524](#)
- readMyMachineType< Field, T >, [522](#)
  - Element, [523](#)
  - Element\_ptr, [523](#)
  - operator(), [523](#)
- readOrRandomMatrixWithRankAndRandomRPM
  - test-nullspace.C, [1092](#)
- readSmsFormat

- FFLAS, [153](#)
- readSprFormat
  - FFLAS, [153](#)
- READWRITE
  - parallel.h, [1032](#)
- Rec\_Initialize
  - benchmark-pluq.C, [798](#)
- RecInt, [403](#)
- recLevel
  - MMHelper< FFPACK::RNSInteger< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, [502](#)
  - MMHelper< FFPACK::RNSIntegerMod< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, [504](#)
  - MMHelper< Field, AlgoTrait, ModeCategories::ConversionTag, Dest >, ParSeqTrait >, [506](#)
  - MMHelper< Field, AlgoTrait, ModeCategories::ConversionTag, ElementCategories::RNSElementTag >, ParSeqTrait >, [508](#)
  - MMHelper< Field, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, [510](#)
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait >, [499](#)
- Recursive, [524](#)
- reduce
  - FFLAS::vectorised, [286](#), [287](#)
  - rns\_double, [527](#)
  - rns\_double\_extended, [538](#)
  - RNSInteger< RNS >, [542](#)
  - RNSIntegerMod< RNS >, [546](#)
- reduce\_modp
  - RNSIntegerMod< RNS >, [548](#), [549](#)
- reduce\_modp\_rnsmajor
  - RNSIntegerMod< RNS >, [549](#)
- ReducedColumnEchelonForm
  - FFPACK, [323](#), [324](#), [374](#)
- ReducedColumnEchelonForm\_modular\_double
  - ffpack.C, [991](#)
  - ffpack\_c.h, [1014](#)
- ReducedColumnEchelonForm\_modular\_float
  - ffpack.C, [992](#)
  - ffpack\_c.h, [1014](#)
- ReducedColumnEchelonForm\_modular\_int32\_t
  - ffpack.C, [993](#)
  - ffpack\_c.h, [1015](#)
- ReducedRowEchelonForm
  - FFPACK, [324](#), [325](#), [373](#)
- ReducedRowEchelonForm2\_modular\_double
  - ffpack\_c.h, [1015](#)
- ReducedRowEchelonForm\_modular\_double
  - ffpack.C, [991](#)
  - ffpack\_c.h, [1014](#)
- ReducedRowEchelonForm\_modular\_float
  - ffpack.C, [992](#)
  - ffpack\_c.h, [1015](#)
- ReducedRowEchelonForm\_modular\_int32\_t
  - ffpack.C, [993](#)
- ffpack\_c.h, [1015](#)
- REF\_modular\_double
  - ffpack\_c.h, [1016](#)
- regression-check.C, [1051](#)
  - check1, [1051](#)
  - check2, [1051](#)
  - check3, [1051](#)
  - check4, [1051](#)
  - checkZeroDimCharpoly, [1051](#)
  - checkZeroDimMinPoly, [1051](#)
  - gf2ModularBalanced, [1051](#)
  - main, [1051](#)
- Residu
  - Bench< Elt >, [411](#)
  - Test< Elt >, [760](#)
- RETURNPARAM
  - parallel.h, [1034](#)
- Ring
  - CheckerImplem\_charpoly< Givaro::ZRing< Givaro::Integer >, Polynomial >, [418](#)
- RNSInteger< RNS >::RandIter, [521](#)
- RNSIntegerMod< RNS >::RandIter, [522](#)
- rnsRandIter< RNS >, [551](#)
- rint< K >, [524](#)
- RNS, [43](#)
  - benchmark-fgemm-rns.C, [785](#)
- rns
  - RNSInteger< RNS >, [541](#)
  - RNSIntegerMod< RNS >, [545](#)
- rns-double-elt.h, [945](#)
- rns-double-recint.inl, [946](#)
  - \_\_FFLASFFPACK\_field\_rns\_double\_recint\_INL, [946](#)
- rns-double.h, [946](#)
  - ROUND\_DOWN, [947](#)
- rns-double.inl, [947](#)
  - \_\_FFLASFFPACK\_field\_rns\_double\_INL, [947](#)
- rns-integer-mod.h, [947](#)
- rns-integer.h, [948](#)
- rns.h, [949](#)
- rns.inl, [949](#)
  - \_\_FFLASFFPACK\_field\_rns\_INL, [949](#)
- rns\_double, [524](#)
  - \_M, [528](#)
  - \_MMi, [528](#)
  - \_Mi, [528](#)
  - \_basis, [528](#)
  - \_basisMax, [528](#)
  - \_crt\_in, [528](#)
  - \_crt\_out, [528](#)
  - \_field\_rns, [528](#)
  - \_invbasis, [528](#)
  - \_ldm, [529](#)
  - \_mi\_sum, [529](#)
  - \_negbasis, [528](#)
  - \_pbits, [529](#)
  - \_size, [528](#)

- BasisElement, 525
- ConstElement\_ptr, 525
- convert, 527, 528
- convert\_transpose, 527
- Element, 525
- Element\_ptr, 525
- init, 526, 527
- init\_transpose, 527
- integer, 525
- ModField, 525
- precompute\_cst, 526
- reduce, 527
- rns\_double, 526
- rns\_double\_elt, 529
  - \_alloc, 530
  - \_ptr, 530
  - \_stride, 530
  - ~rns\_double\_elt, 529
  - operator&, 530
  - rns\_double\_elt, 529, 530
- rns\_double\_elt\_cstptr, 530
  - \_alloc, 533
  - \_ptr, 533
  - \_stride, 533
  - operator!=, 532
  - operator<, 532
  - operator+, 532
  - operator++, 532
  - operator+=, 532
  - operator-, 532
  - operator--, 532
  - operator=, 532
  - operator=, 532
  - operator&, 531, 532
  - operator[], 531, 532
  - operator\*, 531
  - other, 533
  - rns\_double\_elt\_cstptr, 531
- rns\_double\_elt\_ptr, 533
  - \_alloc, 535
  - \_ptr, 535
  - \_stride, 535
  - operator!=, 535
  - operator<, 535
  - operator+, 534
  - operator++, 534
  - operator+=, 535
  - operator-, 535
  - operator--, 534
  - operator=, 535
  - operator=, 535
  - operator&, 534, 535
  - operator[], 534
  - operator\*, 534
  - other, 535
  - rns\_double\_elt\_ptr, 534
- rns\_double\_extended, 536
  - \_M, 539
  - \_MMi, 539
  - \_Mi, 539
  - \_basis, 538
  - \_basisMax, 538
  - \_crt\_in, 539
  - \_crt\_out, 539
  - \_field\_rns, 539
  - \_invbasis, 538
  - \_ldm, 539
  - \_negbasis, 538
  - \_pbits, 539
  - \_size, 539
  - BasisElement, 536
  - ConstElement\_ptr, 537
  - convert, 538
  - Element, 537
  - Element\_ptr, 537
  - init, 537, 538
  - integer, 536
  - ModField, 536
  - precompute\_cst, 537
  - reduce, 538
  - rns\_double\_extended, 537
- RNSElementTag, 539
- RNSInteger
  - RNSInteger< RNS >, 541
- RNSInteger< RNS >, 539
  - \_rns, 543
  - assign, 542
  - BasisElement, 540
  - cardinality, 542
  - characteristic, 541
  - ConstElement\_ptr, 541
  - convert, 542
  - Element, 540
  - Element\_ptr, 541
  - init, 542
  - integer, 540
  - isMOne, 541
  - isOne, 541
  - isZero, 541
  - mOne, 543
  - one, 543
  - reduce, 542
  - rns, 541
  - RNSInteger, 541
  - size, 541
  - write, 542
  - zero, 543
- RNSInteger< RNS >::RandIter, 520
  - operator(), 521
  - RandIter, 521
  - random, 521
  - ring, 521
- RNSIntegerMod
  - RNSIntegerMod< RNS >, 545
- RNSIntegerMod< RNS >, 543
  - \_F, 549



- `_Mi_modp_rns`, 549
- `_RNSdelayed`, 549
- `_iM_modp_rns`, 549
- `_p`, 549
- `_rns`, 549
- `add`, 547
- `areEqual`, 548
- `assign`, 547
- `axpyin`, 547
- `BasisElement`, 545
- `cardinality`, 546
- `characteristic`, 546
- `ConstElement_ptr`, 544
- `convert`, 547
- `delayed`, 545
- `Element`, 544
- `Element_ptr`, 544
- `init`, 546, 547
- `integer`, 545
- `inv`, 548
- `isMOne`, 545
- `isOne`, 545
- `isZero`, 545
- `maxElement`, 546
- `minElement`, 546
- `ModField`, 545
- `mOne`, 549
- `mul`, 547
- `neg`, 547
- `one`, 549
- `reduce`, 546
- `reduce_modp`, 548, 549
- `reduce_modp_rnsmajor`, 549
- `rns`, 545
- `RNSIntegerMod`, 545
- `size`, 545
- `sub`, 547
- `write`, 548
- `write_matrix`, 548
- `write_matrix_long`, 548
- `zero`, 550
- `RNSIntegerMod< RNS >::RandIter`, 521
  - `operator()`, 522
  - `RandIter`, 522
  - `random`, 522
  - `ring`, 522
- `RNSModulus`
  - `FFLAS::CuttingStrategy`, 202
- `rnsRandIter`
  - `rnsRandIter< RNS >`, 550
- `rnsRandIter< RNS >`, 550
  - `operator()`, 550, 551
  - `random`, 550, 551
  - `ring`, 551
  - `rnsRandIter`, 550
- `round`
  - `ScalFunctionsBase< Element, typename enable_if< is_floating_point< Element >::value >::type >`, 551
- `>`, 557
- `ScalFunctionsBase< Element, typename enable_if< is_integral< Element >::value >::type >`, 559
- `Simd128_impl< true, true, false, 2 >`, 571
- `Simd128_impl< true, true, false, 4 >`, 581
- `Simd128_impl< true, true, false, 8 >`, 591
- `Simd128_impl< true, true, true, 2 >`, 600
- `Simd128_impl< true, true, true, 4 >`, 609
- `Simd128_impl< true, true, true, 8 >`, 618
- `Simd256_impl< true, false, true, 8 >`, 628
- `Simd256_impl< true, true, false, 2 >`, 639
- `Simd256_impl< true, true, false, 4 >`, 656
- `Simd256_impl< true, true, false, 8 >`, 666
- `Simd256_impl< true, true, true, 2 >`, 676
- `Simd256_impl< true, true, true, 4 >`, 686, 693
- `Simd256_impl< true, true, true, 8 >`, 702
- `Simd512_impl< true, false, true, 8 >`, 710
- `Simd512_impl< true, true, false, 8 >`, 720
- `Simd512_impl< true, true, true, 8 >`, 730
- `ROUND_DOWN`
  - `fflas_sparse.h`, 884
  - `rns-double.h`, 947
- `Row`, 551
- `row`
  - `Coo< Field >`, 437
  - `Coo< ValT, IdxT >`, 435, 438
  - `Sparse< _Field, SparseMatrix_t::COO >`, 734
  - `Sparse< _Field, SparseMatrix_t::COO_ZO >`, 736
- `rowdim`
  - `StatsMatrix`, 755
- `RowEchelonForm`
  - `FFPACK`, 322, 323, 373
- `RowEchelonForm_modular_double`
  - `ffpack.C`, 991
  - `ffpack_c.h`, 1013
- `RowEchelonForm_modular_float`
  - `ffpack.C`, 992
  - `ffpack_c.h`, 1013
- `RowEchelonForm_modular_int32_t`
  - `ffpack.C`, 993
  - `ffpack_c.h`, 1014
- `RowRankProfile`
  - `FFPACK`, 335, 336, 378
- `RowRankProfile_modular_double`
  - `ffpack.C`, 999
  - `ffpack_c.h`, 1019
- `RowRankProfileSubmatrix`
  - `FFPACK`, 340, 379
- `RowRankProfileSubmatrix_modular_double`
  - `ffpack.C`, 1000
  - `ffpack_c.h`, 1020
- `RowRankProfileSubmatrixIndices`
  - `FFPACK`, 338, 379
- `RowRankProfileSubmatrixIndices_modular_double`
  - `ffpack.C`, 1000
  - `ffpack_c.h`, 1019



- Bench< Elt >, 412
- Test< Elt >, 761
- run\_with\_field
  - benchmark-charpoly.C, 776
  - benchmark-fdot.C, 783
  - benchmark-quasisep.C, 799
  - test-charpoly.C, 1053
  - test-echelon.C, 1057
  - test-fdot.C, 1060
  - test-fgemm-check.C, 1061
  - test-fgemm.C, 1063
  - test-fgemv.C, 1065
  - test-fger.C, 1067
  - test-fgesv.C, 1068
  - test-finit.C, 1069
  - test-fsyr2k.C, 1071
  - test-fsyrk.C, 1073
  - test-fsytrf.C, 1075
  - test-ftrmm.C, 1076
  - test-ftrmv.C, 1077
  - test-ftrsm.C, 1079
  - test-ftrssyr2k.C, 1080
  - test-ftrstr.C, 1081
  - test-ftrsv.C, 1082
  - test-ftrtri.C, 1084
  - test-io.C, 1085
  - test-lu.C, 1089
  - test-minpoly.C, 1091
  - test-nullspace.C, 1093
  - test-quasisep.C, 1096
  - test-rankprofiles.C, 1096
  - test-solve.C, 1101
- run\_with\_Integer
  - test-fdot.C, 1060
- saxpy\_
  - config-blas.h, 818
- ScalAndReduce
  - FFLAS::Protected, 219, 223
- scalar\_t
  - FieldSimd< \_Field >, 451
  - NoSimd< T >, 518
  - Simd128\_impl< true, true, false, 2 >, 564
  - Simd128\_impl< true, true, false, 4 >, 574
  - Simd128\_impl< true, true, false, 8 >, 584
  - Simd128\_impl< true, true, true, 2 >, 594
  - Simd128\_impl< true, true, true, 4 >, 603
  - Simd128\_impl< true, true, true, 8 >, 612
  - Simd256\_impl< true, false, true, 8 >, 623
  - Simd256\_impl< true, true, false, 2 >, 631
  - Simd256\_impl< true, true, false, 4 >, 643
  - Simd256\_impl< true, true, false, 8 >, 660
  - Simd256\_impl< true, true, true, 2 >, 669
  - Simd256\_impl< true, true, true, 4 >, 680
  - Simd256\_impl< true, true, true, 8 >, 696
  - Simd512\_impl< true, false, true, 8 >, 705
  - Simd512\_impl< true, true, false, 8 >, 713
  - Simd512\_impl< true, true, true, 8 >, 723
- ScalFunctions< Element >, 551
  - add, 553
  - addin, 553
  - blend, 556
  - div, 554
  - eq, 555
  - fmadd, 554
  - fmaddin, 554
  - fmsub, 554
  - fmsubin, 554
  - fnmadd, 554
  - fnmaddin, 554
  - genInputs, 552
  - genInputsWithZero, 552
  - greater, 555
  - greater\_eq, 555
  - lesser, 555
  - lesser\_eq, 555
  - mul, 553
  - mulin, 554
  - pack, 556
  - pack\_even, 556
  - pack\_odd, 556
  - sub, 553
  - subin, 553
  - unpackhi, 555
  - unpacklo, 555
  - unpacklohi, 555
  - vand, 552
  - vandnot, 553
  - vectElt, 552
  - vor, 553
  - vxor, 553
  - zero, 552
- ScalFunctionsBase< Element, Enable >, 556
- ScalFunctionsBase< Element, typename enable\_if<
  - is\_floating\_point< Element >::value >::type
  - >, 557
  - \_zero, 558
  - blendv, 557
  - ceil, 557
  - cmp\_false, 558
  - cmp\_true, 558
  - floor, 557
  - fma, 557
  - get\_default\_random\_generator, 557
  - round, 557
- ScalFunctionsBase< Element, typename enable\_if<
  - is\_floating\_point< Element >::value >::type
  - >::FloatingPointTestDistribution, 464
  - FloatingPointTestDistribution, 464
  - IntType, 464
  - operator(), 464
- ScalFunctionsBase< Element, typename enable\_if<
  - is\_integral< Element >::value >::type >, 558
  - \_zero, 560
  - cmp\_false, 561
  - cmp\_true, 560
  - fma, 559

- fmaddx, [559](#)
- fmaddxin, [559](#)
- fmsubx, [559](#)
- fmsubxin, [560](#)
- fnmaddx, [560](#)
- fnmaddxin, [560](#)
- get\_default\_random\_generator, [559](#)
- mulhi, [559](#)
- mullo, [559](#)
- mulx, [559](#)
- round, [559](#)
- sll, [560](#)
- sra, [560](#)
- srl, [560](#)
- scalp
  - FFLAS::vectorised, [287](#), [288](#)
  - FFLAS::vectorised::unswitch, [290](#)
- schedule\_bini.inl, [838](#)
  - \_\_FFLASFFPACK\_fgemm\_bini\_INL, [839](#)
- schedule\_winograd.inl, [839](#)
  - \_\_FFLASFFPACK\_fgemm\_winograd\_INL, [839](#)
- schedule\_winograd\_acc.inl, [839](#)
  - \_\_FFLASFFPACK\_fgemm\_winograd\_acc\_INL, [840](#)
- schedule\_winograd\_acc\_ip.inl, [840](#)
  - \_\_FFLASFFPACK\_fgemm\_winograd\_acc\_ip\_INL, [841](#)
- schedule\_winograd\_ip.inl, [841](#)
  - \_\_FFLASFFPACK\_fgemm\_winograd\_ip\_INL, [841](#)
- scopy\_
  - config-blas.h, [820](#)
- sdot\_
  - config-blas.h, [819](#)
- second\_component
  - Compose< H1, H2 >, [428](#)
- Self
  - Coo< ValT, IdxT >, [434](#), [437](#)
- Self\_t
  - MMHelper< FFPACK::RNSInteger< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, [501](#)
  - MMHelper< FFPACK::RNSIntegerMod< E >, AlgoTrait, ModeCategories::DefaultTag, ParSeqTrait >, [503](#)
  - MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< Simd256\_impl< true, true, true, 2 >, ParSeqTrait >, [505](#)
  - MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo< Simd256\_impl< true, true, true, 8 >, ElementCategories::RNSElementTag >, ParSeqTrait >, [507](#)
  - MMHelper< Field, AlgoTrait, ModeCategories::DefaultTag, Simd512\_impl< true, true, true, 8 >, ParSeqTrait >, [509](#)
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeqTrait >, [497](#)
  - Sparse< \_Field, SparseMatrix\_t::HYB\_ZO >, [749](#)
- SELL
  - FFLAS, [76](#)
- sell.h, [908](#)
- sell\_pspmv.inl, [908](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_sell\_pspmv\_INL, [908](#)
- sell\_spmv.inl, [909](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_sell\_spmv\_INL, [909](#)
- sell\_utils.inl, [909](#)
  - \_\_FFLASFFPACK\_fflas\_sparse\_sell\_utils\_INL, [910](#)
- SELL\_ZO
  - FFLAS, [76](#)
- Sequential, [561](#)
  - numthreads, [561](#)
  - operator<<, [561](#)
  - Sequential, [561](#)
- set
  - Simd128\_impl< true, true, false, 2 >, [564](#), [567](#)
  - Simd128\_impl< true, true, false, 4 >, [575](#), [577](#)
  - Simd128\_impl< true, true, false, 8 >, [585](#), [587](#)
  - Simd128\_impl< true, true, true, 2 >, [595](#)
  - Simd128\_impl< true, true, true, 4 >, [604](#)
  - Simd128\_impl< true, true, true, 8 >, [613](#)
  - Simd256\_impl< true, false, true, 8 >, [624](#)
  - Simd256\_impl< true, true, false, 2 >, [632](#), [635](#)
  - Simd256\_impl< true, true, false, 4 >, [644](#), [646](#), [649](#)
  - Simd256\_impl< true, true, false, 8 >, [660](#), [663](#)
  - Simd256\_impl< true, true, true, 2 >, [670](#)
  - Simd256\_impl< true, true, true, 4 >, [681](#), [687](#)
  - Simd256\_impl< true, true, true, 8 >, [696](#)
  - Simd512\_impl< true, false, true, 8 >, [706](#)
  - Simd512\_impl< true, true, false, 8 >, [713](#), [716](#)
  - Simd512\_impl< true, true, true, 8 >, [724](#)
- set1
  - Simd128\_impl< true, true, false, 2 >, [564](#), [567](#)
  - Simd128\_impl< true, true, false, 4 >, [575](#), [577](#)
  - Simd128\_impl< true, true, false, 8 >, [585](#), [587](#)
  - Simd128\_impl< true, true, true, 2 >, [595](#)
  - Simd128\_impl< true, true, true, 4 >, [604](#)
  - Simd128\_impl< true, true, true, 8 >, [613](#)
  - Simd256\_impl< true, false, true, 8 >, [624](#)
  - Simd256\_impl< true, true, false, 2 >, [632](#), [634](#)
  - Simd256\_impl< true, true, false, 4 >, [644](#), [646](#), [649](#)
  - Simd256\_impl< true, true, false, 8 >, [660](#), [663](#)
  - Simd256\_impl< true, true, true, 2 >, [670](#)
  - Simd256\_impl< true, true, true, 4 >, [681](#), [687](#)
  - Simd256\_impl< true, true, true, 8 >, [696](#)
  - Simd512\_impl< true, false, true, 8 >, [706](#)
  - Simd512\_impl< true, true, false, 8 >, [713](#), [716](#)
  - Simd512\_impl< true, true, true, 8 >, [724](#)
- set\_numthreads
  - Parallel< C, P >, [520](#)
- SET\_THREADS
  - parallel.h, [1032](#)
- setErrorStream
  - Failure, [449](#)
- setNorm

- MMHelper< FFPACK::RNSInteger< E >, Algo-  
Trait, ModeCategories::DefaultTag, ParSeq-  
Trait >, [502](#)
- MMHelper< FFPACK::RNSIntegerMod< E >, Algo-  
Trait, ModeCategories::DefaultTag, ParSeq-  
Trait >, [504](#)
- MMHelper< Field, AlgoTrait, ModeCategories::ConvertTo<  
ElementCategories::RNSElementTag >, ParSeq-  
Trait >, [508](#)
- setOutBounds
  - MMHelper< Field, AlgoTrait, ModeTrait, ParSeq-  
Trait >, [498](#)
- sgemm\_
  - config-blas.h, [822](#)
- sgemv\_
  - config-blas.h, [819](#)
- sger\_
  - config-blas.h, [820](#)
- shuffle
  - Simd128\_impl< true, true, false, 2 >, [568](#)
  - Simd128\_impl< true, true, false, 4 >, [578](#)
  - Simd128\_impl< true, true, false, 8 >, [588](#)
  - Simd128\_impl< true, true, true, 2 >, [596](#)
  - Simd128\_impl< true, true, true, 4 >, [605](#)
  - Simd128\_impl< true, true, true, 8 >, [614](#)
  - Simd256\_impl< true, true, false, 2 >, [636](#)
  - Simd256\_impl< true, true, false, 4 >, [651](#)
  - Simd256\_impl< true, true, false, 8 >, [664](#)
  - Simd256\_impl< true, true, true, 2 >, [671](#)
  - Simd256\_impl< true, true, true, 4 >, [682](#), [688](#)
  - Simd256\_impl< true, true, true, 8 >, [698](#)
  - Simd512\_impl< true, false, true, 8 >, [706](#)
  - Simd512\_impl< true, true, false, 8 >, [717](#)
  - Simd512\_impl< true, true, true, 8 >, [725](#)
- shuffle\_twice
  - Simd256\_impl< true, true, false, 4 >, [650](#), [651](#)
  - Simd256\_impl< true, true, true, 4 >, [682](#), [688](#)
- sigma
  - Sparse< \_Field, SparseMatrix\_t::SELL >, [751](#)
  - Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, [753](#)
- signbits
  - Simd128\_impl< true, true, false, 8 >, [591](#)
  - Simd128\_impl< true, true, true, 8 >, [619](#)
  - Simd256\_impl< true, true, false, 8 >, [667](#)
  - Simd256\_impl< true, true, true, 8 >, [702](#)
  - Simd512\_impl< true, true, false, 8 >, [720](#)
  - Simd512\_impl< true, true, true, 8 >, [730](#)
- Simd
  - fflas\_simd.h, [872](#)
- simd
  - FieldSimd< \_Field >, [451](#)
- SIMD wrapper, [42](#)
- simd.doxy, [872](#)
- Simd128
  - simd128.inl, [873](#)
- simd128.inl, [872](#)
  - \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_INL,  
[872](#)
  - Simd128, [873](#)
  - simd128\_double.inl, [873](#)
    - \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_double\_INL,  
[873](#)
  - simd128\_float.inl, [873](#)
    - \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_float\_INL,  
[873](#)
  - Simd128\_impl< ArithType, Int, Signed, Size >, [562](#)
  - Simd128\_impl< true, false, true, 4 >, [562](#)
  - Simd128\_impl< true, false, true, 8 >, [562](#)
  - Simd128\_impl< true, true, false, 2 >, [562](#)
    - add, [569](#)
    - addin, [569](#)
    - aligned\_allocator, [564](#)
    - aligned\_vector, [564](#)
    - alignment, [572](#)
    - blend, [569](#)
    - compliant, [567](#)
    - eq, [571](#)
    - fmadd, [570](#)
    - fmaddin, [570](#)
    - fmaddx, [566](#)
    - fmaddxin, [566](#)
    - fmsub, [570](#)
    - fmsubin, [571](#)
    - fmsubx, [566](#)
    - fmsubxin, [567](#)
    - fnmadd, [570](#)
    - fnmaddin, [570](#)
    - fnmaddx, [566](#)
    - fnmaddxin, [566](#)
    - gather, [565](#), [567](#)
    - greater, [565](#)
    - greater\_eq, [566](#)
    - hadd\_to\_scal, [567](#)
    - is\_same\_element, [564](#)
    - lesser, [565](#)
    - lesser\_eq, [566](#)
    - load, [565](#), [567](#)
    - loadu, [565](#), [567](#)
    - mod, [571](#)
    - mul, [570](#)
    - mulhi, [566](#)
    - mullo, [570](#)
    - mulx, [566](#)
    - pack, [569](#)
    - pack\_even, [569](#)
    - pack\_odd, [569](#)
    - round, [571](#)
    - scalar\_t, [564](#)
    - set, [564](#), [567](#)
    - set1, [564](#), [567](#)
    - shuffle, [568](#)
    - sll, [568](#)
    - sll128, [571](#)
    - sra, [565](#)
    - srl, [568](#)
    - srl128, [571](#)

- store, [565](#), [568](#)
- storeu, [565](#), [568](#)
- stream, [565](#), [568](#)
- sub, [570](#)
- subin, [570](#)
- transpose, [569](#)
- type\_string, [564](#)
- unpackhi, [568](#)
- unpackhi\_intrinsic, [568](#)
- unpacklo, [568](#)
- unpacklo\_intrinsic, [568](#)
- unpacklohi, [569](#)
- valid, [567](#)
- vand, [571](#)
- vandnot, [572](#)
- vect\_size, [572](#)
- vect\_t, [564](#)
- vor, [571](#)
- vxor, [572](#)
- zero, [571](#)
- Simd128\_impl< true, true, false, 2 >::Converter, [429](#)
- t, [429](#)
- v, [429](#)
- Simd128\_impl< true, true, false, 4 >, [572](#)
- add, [579](#)
- addin, [579](#)
- aligned\_allocator, [574](#)
- aligned\_vector, [574](#)
- alignment, [582](#)
- blend, [579](#)
- compliant, [577](#)
- eq, [581](#)
- fmadd, [580](#)
- fmaddin, [580](#)
- fmaddx, [576](#)
- fmaddxin, [576](#)
- fmsub, [580](#)
- fmsubin, [581](#)
- fmsubx, [577](#)
- fmsubxin, [577](#)
- fnmadd, [580](#)
- fnmaddin, [580](#)
- fnmaddx, [576](#)
- fnmaddxin, [576](#)
- gather, [575](#), [577](#)
- greater, [575](#)
- greater\_eq, [576](#)
- hadd\_to\_scal, [577](#)
- is\_same\_element, [574](#)
- lesser, [576](#)
- lesser\_eq, [576](#)
- load, [575](#), [577](#)
- loadu, [575](#), [578](#)
- mod, [581](#)
- mul, [580](#)
- mulhi, [576](#)
- mullo, [580](#)
- mulx, [576](#)
- pack, [579](#)
- pack\_even, [579](#)
- pack\_odd, [579](#)
- round, [581](#)
- scalar\_t, [574](#)
- set, [575](#), [577](#)
- set1, [575](#), [577](#)
- shuffle, [578](#)
- sll, [578](#)
- sll128, [581](#)
- sra, [575](#)
- srl, [578](#)
- srl128, [581](#)
- store, [575](#), [578](#)
- storeu, [575](#), [578](#)
- stream, [575](#), [578](#)
- sub, [580](#)
- subin, [580](#)
- transpose, [579](#)
- type\_string, [575](#)
- unpackhi, [579](#)
- unpackhi\_intrinsic, [578](#)
- unpacklo, [578](#)
- unpacklo\_intrinsic, [578](#)
- unpacklohi, [579](#)
- valid, [577](#)
- vand, [581](#)
- vandnot, [582](#)
- vect\_size, [582](#)
- vect\_t, [574](#)
- vor, [581](#)
- vxor, [582](#)
- zero, [581](#)
- Simd128\_impl< true, true, false, 4 >::Converter, [429](#)
- t, [429](#)
- v, [429](#)
- Simd128\_impl< true, true, false, 8 >, [582](#)
- add, [590](#)
- addin, [590](#)
- aligned\_allocator, [584](#)
- aligned\_vector, [584](#)
- alignment, [592](#)
- blend, [589](#)
- compliant, [587](#)
- eq, [591](#)
- fmadd, [590](#)
- fmaddin, [590](#)
- fmaddx, [586](#)
- fmaddxin, [586](#)
- fmsub, [591](#)
- fmsubin, [591](#)
- fmsubx, [587](#)
- fmsubxin, [587](#)
- fnmadd, [590](#)
- fnmaddin, [590](#)
- fnmaddx, [586](#)
- fnmaddxin, [587](#)
- gather, [585](#), [587](#)

- get, [588](#)
- greater, [585](#)
- greater\_eq, [586](#)
- hadd\_to\_scal, [587](#)
- is\_same\_element, [584](#)
- lesser, [586](#)
- lesser\_eq, [586](#)
- load, [585](#), [588](#)
- loadu, [585](#), [588](#)
- mask\_high, [591](#)
- mod, [591](#)
- mul, [590](#)
- mulhi, [586](#)
- mulhi\_fast, [591](#)
- mullo, [586](#)
- mulx, [586](#)
- pack, [589](#)
- pack\_even, [589](#)
- pack\_odd, [589](#)
- round, [591](#)
- scalar\_t, [584](#)
- set, [585](#), [587](#)
- set1, [585](#), [587](#)
- shuffle, [588](#)
- signbits, [591](#)
- sll, [588](#)
- sll128, [592](#)
- sra, [585](#)
- srl, [588](#)
- srl128, [592](#)
- store, [585](#), [588](#)
- storeu, [585](#), [588](#)
- stream, [585](#), [588](#)
- sub, [590](#)
- subin, [590](#)
- transpose, [589](#)
- type\_string, [585](#)
- unpackhi, [589](#)
- unpackhi\_intrinsic, [589](#)
- unpacklo, [589](#)
- unpacklo\_intrinsic, [588](#)
- unpacklohi, [589](#)
- valid, [587](#)
- vand, [592](#)
- vandnot, [592](#)
- vect\_size, [592](#)
- vect\_t, [584](#)
- vor, [592](#)
- vxor, [592](#)
- zero, [591](#)
- Simd128\_impl< true, true, false, 8 >::Converter, [429](#)
- t, [429](#)
- v, [429](#)
- Simd128\_impl< true, true, true, 2 >, [592](#)
- add, [597](#)
- addin, [598](#)
- aligned\_allocator, [594](#)
- aligned\_vector, [595](#)
- alignment, [601](#)
- blend, [597](#)
- compliant, [595](#)
- eq, [600](#)
- fmadd, [598](#)
- fmaddin, [598](#)
- fmaddx, [598](#)
- fmaddxin, [599](#)
- fmsub, [599](#)
- fmsubin, [599](#)
- fmsubx, [599](#)
- fmsubxin, [600](#)
- fnmadd, [599](#)
- fnmaddin, [599](#)
- fnmaddx, [599](#)
- fnmaddxin, [599](#)
- gather, [595](#)
- greater, [600](#)
- greater\_eq, [600](#)
- hadd\_to\_scal, [600](#)
- is\_same\_element, [595](#)
- lesser, [600](#)
- lesser\_eq, [600](#)
- load, [595](#)
- loadu, [595](#)
- mod, [600](#)
- mul, [598](#)
- mulhi, [598](#)
- mullo, [598](#)
- mulx, [598](#)
- pack, [597](#)
- pack\_even, [597](#)
- pack\_odd, [597](#)
- round, [600](#)
- scalar\_t, [594](#)
- set, [595](#)
- set1, [595](#)
- shuffle, [596](#)
- sll, [596](#)
- sll128, [601](#)
- sra, [596](#)
- srl, [596](#)
- srl128, [601](#)
- store, [596](#)
- storeu, [596](#)
- stream, [596](#)
- sub, [598](#)
- subin, [598](#)
- transpose, [597](#)
- type\_string, [595](#)
- unpackhi, [597](#)
- unpackhi\_intrinsic, [596](#)
- unpacklo, [596](#)
- unpacklo\_intrinsic, [596](#)
- unpacklohi, [597](#)
- valid, [595](#)
- vand, [601](#)
- vandnot, [601](#)

- vect\_size, 601
- vect\_t, 594
- vor, 601
- vxor, 601
- zero, 601
- Simd128\_impl< true, true, true, 2 >::Converter, 430
  - t, 430
  - v, 430
- Simd128\_impl< true, true, true, 4 >, 602
  - add, 606
  - addin, 606
  - aligned\_allocator, 603
  - aligned\_vector, 604
  - alignment, 610
  - blend, 606
  - compliant, 604
  - eq, 609
  - fmadd, 607
  - fmaddin, 607
  - fmaddx, 607
  - fmaddxin, 607
  - fmsub, 608
  - fmsubin, 608
  - fmsubx, 608
  - fmsubxin, 608
  - fnmadd, 608
  - fnmaddin, 608
  - fnmaddx, 608
  - fnmaddxin, 608
  - gather, 604
  - greater, 609
  - greater\_eq, 609
  - hadd\_to\_scal, 609
  - is\_same\_element, 604
  - lesser, 609
  - lesser\_eq, 609
  - load, 604
  - loadu, 604
  - mod, 609
  - mul, 607
  - mulhi, 607
  - mullo, 607
  - mulx, 607
  - pack, 606
  - pack\_even, 606
  - pack\_odd, 606
  - round, 609
  - scalar\_t, 603
  - set, 604
  - set1, 604
  - shuffle, 605
  - sll, 605
  - sll128, 609
  - sra, 605
  - srl, 605
  - srl128, 610
  - store, 604
  - storeu, 605
  - stream, 605
  - sub, 606
  - subin, 607
  - transpose, 606
  - type\_string, 604
  - unpackhi, 605
  - unpackhi\_intrinsic, 605
  - unpacklo, 605
  - unpacklo\_intrinsic, 605
  - unpacklohi, 606
  - valid, 604
  - vand, 610
  - vandnot, 610
  - vect\_size, 610
  - vect\_t, 603
  - vor, 610
  - vxor, 610
  - zero, 609
- Simd128\_impl< true, true, true, 4 >::Converter, 430
  - t, 430
  - v, 430
- Simd128\_impl< true, true, true, 8 >, 610
  - add, 615
  - addin, 616
  - aligned\_allocator, 613
  - aligned\_vector, 613
  - alignment, 620
  - blend, 615
  - compliant, 613
  - eq, 618
  - fmadd, 616
  - fmaddin, 616
  - fmaddx, 616
  - fmaddxin, 617
  - fmsub, 617
  - fmsubin, 617
  - fmsubx, 617
  - fmsubxin, 618
  - fnmadd, 617
  - fnmaddin, 617
  - fnmaddx, 617
  - fnmaddxin, 617
  - gather, 613
  - get, 613
  - greater, 618
  - greater\_eq, 618
  - hadd\_to\_scal, 618
  - is\_same\_element, 613
  - lesser, 618
  - lesser\_eq, 618
  - load, 613
  - loadu, 614
  - mask\_high, 618
  - mod, 618
  - mul, 616
  - mulhi, 616
  - mulhi\_fast, 618
  - mullo, 616

- mulx, [616](#)
- pack, [615](#)
- pack\_even, [615](#)
- pack\_odd, [615](#)
- round, [618](#)
- scalar\_t, [612](#)
- set, [613](#)
- set1, [613](#)
- shuffle, [614](#)
- signbits, [619](#)
- sll, [614](#)
- sll128, [619](#)
- sra, [614](#)
- srl, [614](#)
- srl128, [619](#)
- store, [614](#)
- storeu, [614](#)
- stream, [614](#)
- sub, [616](#)
- subin, [616](#)
- transpose, [615](#)
- type\_string, [613](#)
- unpackhi, [615](#)
- unpackhi\_intrinsic, [614](#)
- unpacklo, [615](#)
- unpacklo\_intrinsic, [614](#)
- unpacklohi, [615](#)
- valid, [613](#)
- vand, [619](#)
- vandnot, [619](#)
- vect\_size, [620](#)
- vect\_t, [612](#)
- vor, [619](#)
- vxor, [619](#)
- zero, [619](#)
- Simd128\_impl< true, true, true, 8 >::Converter, [430](#)
- t, [430](#)
- v, [430](#)
- simd128\_int16.inl, [873](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_int16\_INL, [874](#)
- simd128\_int32.inl, [874](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_int32\_INL, [874](#)
- simd128\_int64.inl, [874](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd128\_int64\_INL, [875](#)
- vect\_t, [875](#)
- Simd128i\_base, [620](#)
- sll128, [620](#)
- srl128, [620](#)
- vand, [621](#)
- vandnot, [621](#)
- vect\_t, [620](#)
- vor, [621](#)
- vxor, [621](#)
- zero, [620](#)
- Simd256
- simd256.inl, [875](#)
- simd256.inl, [875](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_INL, [875](#)
- Simd256, [875](#)
- simd256\_double.inl, [875](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_double\_INL, [876](#)
- simd256\_float.inl, [876](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_float\_INL, [876](#)
- Simd256\_impl< ArithType, Int, Signed, Size >, [621](#)
- Simd256\_impl< true, false, true, 4 >, [621](#)
- Simd256\_impl< true, false, true, 8 >, [622](#)
- add, [626](#)
- addin, [626](#)
- aligned\_allocator, [623](#)
- aligned\_vector, [623](#)
- alignment, [629](#)
- blend, [626](#)
- blendv, [626](#)
- ceil, [628](#)
- compliant, [624](#)
- div, [626](#)
- eq, [627](#)
- floor, [628](#)
- fmadd, [627](#)
- fmaddin, [627](#)
- fmsub, [627](#)
- fmsubin, [627](#)
- fnmadd, [627](#)
- fnmaddin, [627](#)
- gather, [624](#)
- greater, [628](#)
- greater\_eq, [628](#)
- hadd, [628](#)
- hadd\_to\_scal, [629](#)
- is\_same\_element, [623](#)
- lesser, [627](#)
- lesser\_eq, [627](#)
- load, [624](#)
- loadu, [624](#)
- mod, [629](#)
- mul, [626](#)
- mulin, [626](#)
- pack, [625](#)
- pack\_even, [625](#)
- pack\_odd, [625](#)
- round, [628](#)
- scalar\_t, [623](#)
- set, [624](#)
- set1, [624](#)
- store, [624](#)
- storeu, [624](#)
- stream, [624](#)
- sub, [626](#)
- subin, [626](#)
- transpose, [625](#)

- type\_string, 623
- unpackhi, 625
- unpackhi\_intrinsic, 625
- unpacklo, 625
- unpacklo\_intrinsic, 625
- unpacklohi, 625
- valid, 623
- vand, 628
- vandnot, 628
- vect\_size, 629
- vect\_t, 623
- vor, 628
- vxor, 628
- zero, 624
- Simd256\_impl< true, false, true, 8 >::Converter, 431
  - t, 431
  - v, 431
- Simd256\_impl< true, true, false, 2 >, 629
  - add, 637
  - addin, 637
  - aligned\_allocator, 631
  - aligned\_vector, 631
  - alignment, 639
  - blend, 637
  - compliant, 634
  - eq, 638
  - fmadd, 638
  - fmaddin, 638
  - fmaddx, 633
  - fmaddxin, 634
  - fmsub, 638
  - fmsubin, 638
  - fmsubx, 634
  - fmsubxin, 634
  - fnmadd, 638
  - fnmaddin, 638
  - fnmaddx, 634
  - fnmaddxin, 634
  - gather, 632, 635
  - greater, 633
  - greater\_eq, 633
  - hadd\_to\_scal, 634
  - half\_t, 632
  - is\_same\_element, 631
  - lesser, 633
  - lesser\_eq, 633
  - load, 632, 635
  - loadu, 632, 635
  - mod, 639
  - mul, 638
  - mulhi, 633
  - mullo, 638
  - mulx, 633
  - pack, 636
  - pack\_even, 636
  - pack\_odd, 636
  - round, 639
  - scalar\_t, 631
  - set, 632, 635
  - set1, 632, 634
  - shuffle, 636
  - simdHalf, 631
  - sll, 635
  - sra, 633
  - srl, 636
  - store, 632, 635
  - storeu, 633, 635
  - stream, 633, 635
  - sub, 637
  - subin, 637
  - transpose, 637
  - type\_string, 632
  - unpackhi, 636
  - unpackhi\_intrinsic, 636
  - unpacklo, 636
  - unpacklo\_intrinsic, 636
  - unpacklohi, 636
  - valid, 634
  - vect\_size, 639
  - vect\_t, 632
  - zero, 639
- Simd256\_impl< true, true, false, 2 >::Converter, 431
  - t, 431
  - v, 431
- Simd256\_impl< true, true, false, 4 >, 639
  - add, 653
  - addin, 654
  - aligned\_allocator, 643
  - aligned\_vector, 643
  - alignment, 657
  - blend, 653
  - compliant, 649
  - eq, 656
  - fmadd, 655
  - fmaddin, 655
  - fmaddx, 645, 648
  - fmaddxin, 645, 648
  - fmsub, 656
  - fmsubin, 656
  - fmsubx, 646, 648
  - fmsubxin, 646, 648
  - fnmadd, 655
  - fnmaddin, 655
  - fnmaddx, 645, 648
  - fnmaddxin, 646, 648
  - gather, 644, 646, 649
  - greater, 645, 647
  - greater\_eq, 645, 647
  - hadd\_to\_scal, 646, 648
  - half\_t, 643
  - is\_same\_element, 643
  - lesser, 645, 647
  - lesser\_eq, 645, 647
  - load, 644, 646, 650
  - loadu, 644, 647, 650
  - mod, 656, 657



- mul, [654](#)
- mulhi, [645](#), [647](#)
- mullo, [654](#)
- mulx, [645](#), [648](#)
- pack, [652](#)
- pack\_even, [652](#)
- pack\_odd, [652](#)
- round, [656](#)
- scalar\_t, [643](#)
- set, [644](#), [646](#), [649](#)
- set1, [644](#), [646](#), [649](#)
- shuffle, [651](#)
- shuffle\_twice, [650](#), [651](#)
- simdHalf, [643](#)
- sll, [650](#)
- sra, [644](#), [647](#)
- srl, [650](#)
- store, [644](#), [647](#), [650](#)
- storeu, [644](#), [647](#), [650](#)
- stream, [644](#), [647](#), [650](#)
- sub, [654](#)
- subin, [654](#)
- transpose, [653](#)
- type\_string, [644](#), [646](#)
- unpackhi, [651](#), [652](#)
- unpackhi\_intrinsic, [651](#)
- unpacklo, [651](#)
- unpacklo\_intrinsic, [651](#)
- unpacklohi, [652](#)
- valid, [648](#), [649](#)
- vand, [657](#)
- vandnot, [657](#)
- vect\_size, [657](#)
- vect\_t, [643](#)
- vor, [657](#)
- vxor, [657](#)
- zero, [657](#)
- Simd256\_impl< true, true, false, 4 >::Converter, [431](#)
  - t, [431](#)
  - v, [431](#)
- Simd256\_impl< true, true, false, 8 >, [658](#)
  - add, [665](#)
  - addin, [665](#)
  - aligned\_allocator, [660](#)
  - aligned\_vector, [660](#)
  - alignment, [667](#)
  - blend, [665](#)
  - compliant, [663](#)
  - eq, [666](#)
  - fmadd, [666](#)
  - fmaddin, [666](#)
  - fmaddx, [662](#)
  - fmaddxin, [662](#)
  - fmsub, [666](#)
  - fmsubin, [666](#)
  - fmsubx, [662](#)
  - fmsubxin, [662](#)
  - fnmadd, [666](#)
  - fnmaddin, [666](#)
  - fnmaddx, [662](#)
  - fnmaddxin, [662](#)
  - gather, [660](#), [663](#)
  - get, [663](#)
  - greater, [661](#)
  - greater\_eq, [661](#)
  - hadd\_to\_scal, [663](#)
  - half\_t, [660](#)
  - is\_same\_element, [660](#)
  - lesser, [661](#)
  - lesser\_eq, [661](#)
  - load, [661](#), [663](#)
  - loadu, [661](#), [663](#)
  - mask\_high, [667](#)
  - mod, [667](#)
  - mul, [666](#)
  - mulhi, [662](#)
  - mulhi\_fast, [667](#)
  - mullo, [662](#)
  - mulx, [662](#)
  - pack, [665](#)
  - pack\_even, [665](#)
  - pack\_odd, [665](#)
  - round, [666](#)
  - scalar\_t, [660](#)
  - set, [660](#), [663](#)
  - set1, [660](#), [663](#)
  - shuffle, [664](#)
  - signbits, [667](#)
  - simdHalf, [660](#)
  - sll, [664](#)
  - sra, [661](#)
  - srl, [664](#)
  - store, [661](#), [663](#)
  - storeu, [661](#), [664](#)
  - stream, [661](#), [664](#)
  - sub, [665](#)
  - subin, [665](#)
  - transpose, [665](#)
  - type\_string, [660](#)
  - unpackhi, [664](#)
  - unpackhi\_intrinsic, [664](#)
  - unpacklo, [664](#)
  - unpacklo\_intrinsic, [664](#)
  - unpacklohi, [664](#)
  - valid, [663](#)
  - vect\_size, [667](#)
  - vect\_t, [660](#)
  - zero, [667](#)
- Simd256\_impl< true, true, false, 8 >::Converter, [432](#)
  - t, [432](#)
  - v, [432](#)
- Simd256\_impl< true, true, true, 2 >, [667](#)
  - add, [673](#)
  - addin, [673](#)
  - aligned\_allocator, [670](#)
  - aligned\_vector, [670](#)

- alignment, 676
- blend, 673
- compliant, 670
- eq, 675
- fmadd, 674
- fmaddin, 674
- fmaddx, 674
- fmaddxin, 674
- fmsub, 675
- fmsubin, 675
- fmsubx, 675
- fmsubxin, 675
- fnmadd, 674
- fnmaddin, 674
- fnmaddx, 674
- fnmaddxin, 675
- gather, 670
- greater, 675
- greater\_eq, 675
- hadd\_to\_scal, 676
- half\_t, 669
- is\_same\_element, 670
- lesser, 675
- lesser\_eq, 676
- load, 671
- loadu, 671
- mod, 676
- mul, 673
- mulhi, 674
- mullo, 673
- mulx, 674
- pack, 672
- pack\_even, 672
- pack\_odd, 672
- round, 676
- scalar\_t, 669
- set, 670
- set1, 670
- shuffle, 671
- simdHalf, 669
- sll, 671
- sra, 671
- srl, 671
- store, 671
- storeu, 671
- stream, 671
- sub, 673
- subin, 673
- transpose, 672
- type\_string, 670
- unpackhi, 672
- unpackhi\_intrinsic, 672
- unpacklo, 672
- unpacklo\_intrinsic, 671
- unpacklohi, 672
- valid, 670
- vect\_size, 676
- vect\_t, 669
- zero, 676
- Simd256\_impl< true, true, true, 2 >::Converter, 432
  - t, 432
  - v, 432
- Simd256\_impl< true, true, true, 4 >, 676
  - add, 684, 690
  - addin, 684, 690
  - aligned\_allocator, 680
  - aligned\_vector, 680, 681
  - alignment, 693
  - blend, 683, 690
  - compliant, 681, 687
  - eq, 686, 692
  - fmadd, 684, 691
  - fmaddin, 684, 691
  - fmaddx, 685, 691
  - fmaddxin, 685, 691
  - fmsub, 685, 692
  - fmsubin, 685, 692
  - fmsubx, 686, 692
  - fmsubxin, 686, 692
  - fnmadd, 685, 691
  - fnmaddin, 685, 691
  - fnmaddx, 685, 691
  - fnmaddxin, 685, 691
  - gather, 681, 687
  - greater, 686, 692
  - greater\_eq, 686, 692
  - hadd\_to\_scal, 686, 692
  - half\_t, 680
  - is\_same\_element, 680, 681
  - lesser, 686, 692
  - lesser\_eq, 686, 692
  - load, 681, 687
  - loadu, 681, 688
  - mod, 686, 693
  - mul, 684, 690
  - mulhi, 684, 690
  - mullo, 684, 690
  - mulx, 684, 690
  - pack, 683, 689
  - pack\_even, 683, 689
  - pack\_odd, 683, 689
  - round, 686, 693
  - scalar\_t, 680
  - set, 681, 687
  - set1, 681, 687
  - shuffle, 682, 688
  - shuffle\_twice, 682, 688
  - simdHalf, 680
  - sll, 682, 688
  - sra, 682, 688
  - srl, 682, 688
  - store, 682, 688
  - storeu, 682, 688
  - stream, 682, 688
  - sub, 684, 690
  - subin, 684, 690

- transpose, [683](#), [689](#)
- type\_string, [681](#), [687](#)
- unpackhi, [683](#), [689](#)
- unpackhi\_intrinsic, [682](#), [689](#)
- unpacklo, [683](#), [689](#)
- unpacklo\_intrinsic, [682](#), [688](#)
- unpacklohi, [683](#), [689](#)
- valid, [681](#), [687](#)
- vand, [693](#)
- vandnot, [693](#)
- vect\_size, [693](#)
- vect\_t, [680](#)
- vor, [693](#)
- vxor, [693](#)
- zero, [693](#)
- Simd256\_impl< true, true, true, 4 >::Converter, [432](#)
- t, [432](#)
- v, [432](#)
- Simd256\_impl< true, true, true, 8 >, [694](#)
- add, [699](#)
- addin, [699](#)
- aligned\_allocator, [696](#)
- aligned\_vector, [696](#)
- alignment, [702](#)
- blend, [699](#)
- compliant, [696](#)
- eq, [701](#)
- fmadd, [700](#)
- fmaddin, [700](#)
- fmaddx, [700](#)
- fmaddxin, [700](#)
- fmsub, [701](#)
- fmsubin, [701](#)
- fmsubx, [701](#)
- fmsubxin, [701](#)
- fnmadd, [700](#)
- fnmaddin, [700](#)
- fnmaddx, [700](#)
- fnmaddxin, [700](#)
- gather, [697](#)
- get, [697](#)
- greater, [701](#)
- greater\_eq, [701](#)
- hadd\_to\_scal, [702](#)
- half\_t, [696](#)
- is\_same\_element, [696](#)
- lesser, [701](#)
- lesser\_eq, [701](#)
- load, [697](#)
- loadu, [697](#)
- mask\_high, [702](#)
- mod, [702](#)
- mul, [699](#)
- mulhi, [699](#)
- mulhi\_fast, [702](#)
- mullo, [699](#)
- mulx, [700](#)
- pack, [698](#)
- pack\_even, [698](#)
- pack\_odd, [698](#)
- round, [702](#)
- scalar\_t, [696](#)
- set, [696](#)
- set1, [696](#)
- shuffle, [698](#)
- signbits, [702](#)
- simdHalf, [696](#)
- sll, [697](#)
- sra, [697](#)
- srl, [697](#)
- store, [697](#)
- storeu, [697](#)
- stream, [697](#)
- sub, [699](#)
- subin, [699](#)
- transpose, [699](#)
- type\_string, [696](#)
- unpackhi, [698](#)
- unpackhi\_intrinsic, [698](#)
- unpacklo, [698](#)
- unpacklo\_intrinsic, [698](#)
- unpacklohi, [698](#)
- valid, [696](#)
- vect\_size, [702](#)
- vect\_t, [696](#)
- zero, [702](#)
- Simd256\_impl< true, true, true, 8 >::Converter, [433](#)
- t, [433](#)
- v, [433](#)
- simd256\_int16.inl, [876](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_int16\_INL, [877](#)
- simd256\_int32.inl, [877](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_int32\_INL, [877](#)
- simd256\_int64.inl, [877](#)
- \_\_FFLASFFPACK\_fflas\_ffpack\_utils\_simd256\_int64\_INL, [877](#)
- vect\_t, [877](#)
- Simd256fp\_base, [703](#)
- Simd256i\_base, [703](#)
- vect\_t, [703](#)
- zero, [703](#)
- Simd512
- simd512.inl, [878](#)
- simd512.inl, [878](#)
- \_\_FFLASFFPACK\_simd512\_INL, [878](#)
- Simd512, [878](#)
- simd512\_double.inl, [878](#)
- \_\_FFLASFFPACK\_simd512\_double\_INL, [878](#)
- simd512\_float.inl, [879](#)
- \_\_FFLASFFPACK\_simd512\_float\_INL, [879](#)
- Simd512\_impl< ArithType, Int, Signed, Size >, [703](#)
- Simd512\_impl< true, false, true, 4 >, [703](#)
- Simd512\_impl< true, false, true, 8 >, [704](#)
- add, [708](#)

- addin, [708](#)
- aligned\_allocator, [705](#)
- aligned\_vector, [705](#)
- alignment, [710](#)
- blend, [708](#)
- blendv, [708](#)
- ceil, [710](#)
- compliant, [705](#)
- div, [708](#)
- eq, [709](#)
- floor, [710](#)
- fmadd, [708](#)
- fmaddin, [709](#)
- fmsub, [709](#)
- fmsubin, [709](#)
- fnmadd, [709](#)
- fnmaddin, [709](#)
- gather, [706](#)
- greater, [709](#)
- greater\_eq, [710](#)
- hadd, [710](#)
- hadd\_to\_scal, [710](#)
- is\_same\_element, [705](#)
- lesser, [709](#)
- lesser\_eq, [709](#)
- load, [706](#)
- loadu, [706](#)
- mul, [708](#)
- mulin, [708](#)
- pack, [707](#)
- pack\_even, [707](#)
- pack\_odd, [707](#)
- round, [710](#)
- scalar\_t, [705](#)
- set, [706](#)
- set1, [706](#)
- shuffle, [706](#)
- store, [706](#)
- storeu, [706](#)
- stream, [706](#)
- sub, [708](#)
- subin, [708](#)
- transpose, [707](#)
- type\_string, [705](#)
- unpackhi, [707](#)
- unpackhi\_intrinsic, [707](#)
- unpacklo, [707](#)
- unpacklo\_intrinsic, [706](#)
- unpacklohi, [707](#)
- valid, [705](#)
- vect\_size, [710](#)
- vect\_t, [705](#)
- zero, [705](#)
- Simd512\_impl< true, true, false, 8 >, [710](#)
  - add, [719](#)
  - addin, [719](#)
  - aligned\_allocator, [713](#)
  - aligned\_vector, [713](#)
  - alignment, [721](#)
  - blend, [719](#)
  - compliant, [716](#)
  - eq, [720](#)
  - fmadd, [719](#)
  - fmaddin, [719](#)
  - fmaddx, [715](#)
  - fmaddxin, [715](#)
  - fmsub, [720](#)
  - fmsubin, [720](#)
  - fmsubx, [715](#)
  - fmsubxin, [716](#)
  - fnmadd, [719](#)
  - fnmaddin, [720](#)
  - fnmaddx, [715](#)
  - fnmaddxin, [715](#)
  - gather, [713](#), [716](#)
  - greater, [714](#)
  - greater\_eq, [714](#)
  - hadd\_to\_scal, [716](#)
  - half\_t, [713](#)
  - is\_same\_element, [713](#)
  - lesser, [714](#)
  - lesser\_eq, [715](#)
  - load, [714](#), [717](#)
  - loadu, [714](#), [717](#)
  - mask\_high, [720](#)
  - maskstore, [714](#), [717](#)
  - mod, [720](#)
  - mul, [719](#)
  - mulhi, [715](#)
  - mulhi\_fast, [720](#)
  - mullo, [715](#)
  - mulx, [715](#)
  - pack, [718](#)
  - pack\_even, [718](#)
  - pack\_odd, [718](#)
  - round, [720](#)
  - scalar\_t, [713](#)
  - set, [713](#), [716](#)
  - set1, [713](#), [716](#)
  - shuffle, [717](#)
  - signbits, [720](#)
  - simdHalf, [713](#)
  - sll, [717](#)
  - sra, [714](#)
  - srl, [717](#)
  - store, [714](#), [717](#)
  - storeu, [714](#), [717](#)
  - stream, [714](#), [717](#)
  - sub, [719](#)
  - subin, [719](#)
  - transpose, [718](#)
  - type\_string, [713](#)
  - unpackhi, [718](#)
  - unpackhi\_intrinsic, [718](#)
  - unpacklo, [718](#)
  - unpacklo\_intrinsic, [717](#)

- unpacklohi, [718](#)
- valid, [716](#)
- vand, [721](#)
- vandnot, [721](#)
- vect\_size, [721](#)
- vect\_t, [713](#)
- vor, [721](#)
- vxor, [721](#)
- zero, [721](#)
- Simd512\_impl< true, true, false, 8 >::Converter, [433](#)
- t, [433](#)
- v, [433](#)
- Simd512\_impl< true, true, true, 8 >, [721](#)
- add, [727](#)
- addin, [727](#)
- aligned\_allocator, [724](#)
- aligned\_vector, [724](#)
- alignment, [731](#)
- blend, [727](#)
- compliant, [724](#)
- eq, [729](#)
- fmadd, [728](#)
- fmaddin, [728](#)
- fmaddx, [728](#)
- fmaddxin, [728](#)
- fmsub, [729](#)
- fmsubin, [729](#)
- fmsubx, [729](#)
- fmsubxin, [729](#)
- fnmadd, [728](#)
- fnmaddin, [728](#)
- fnmaddx, [728](#)
- fnmaddxin, [728](#)
- gather, [724](#)
- greater, [729](#)
- greater\_eq, [729](#)
- hadd\_to\_scal, [729](#)
- half\_t, [723](#)
- is\_same\_element, [724](#)
- lesser, [729](#)
- lesser\_eq, [729](#)
- load, [725](#)
- loadu, [725](#)
- mask\_high, [730](#)
- maskstore, [725](#)
- mod, [730](#)
- mul, [727](#)
- mulhi, [727](#)
- mulhi\_fast, [730](#)
- mullo, [727](#)
- mulx, [727](#)
- pack, [726](#)
- pack\_even, [726](#)
- pack\_odd, [726](#)
- round, [730](#)
- scalar\_t, [723](#)
- set, [724](#)
- set1, [724](#)
- shuffle, [725](#)
- signbits, [730](#)
- simdHalf, [723](#)
- sll, [725](#)
- sra, [725](#)
- srl, [725](#)
- store, [725](#)
- storeu, [725](#)
- stream, [725](#)
- sub, [727](#)
- subin, [727](#)
- transpose, [726](#)
- type\_string, [724](#)
- unpackhi, [726](#)
- unpackhi\_intrinsic, [726](#)
- unpacklo, [726](#)
- unpacklo\_intrinsic, [726](#)
- unpacklohi, [726](#)
- valid, [724](#)
- vand, [730](#)
- vandnot, [730](#)
- vect\_size, [731](#)
- vect\_t, [723](#)
- vor, [730](#)
- vxor, [730](#)
- zero, [730](#)
- Simd512\_impl< true, true, true, 8 >::Converter, [433](#)
- t, [433](#)
- v, [433](#)
- simd512\_int32.inl, [879](#)
- \_\_FFLASFFPACK\_simd512\_int32\_INL, [879](#)
- simd512\_int64.inl, [879](#)
- \_simd512\_int64\_INL, [880](#)
- vect\_t, [880](#)
- Simd512i\_base, [731](#)
- vand, [732](#)
- vandnot, [732](#)
- vect\_t, [731](#)
- vor, [731](#)
- vxor, [731](#)
- zero, [731](#)
- SIMD\_INT
- fflas\_simd.h, [871](#)
- simd\_modular.inl, [880](#)
- SimdChooser< T, bool, bool >, [732](#)
- SimdChooser< T, false, b >, [732](#)
- value, [732](#)
- SimdChooser< T, true, false >, [732](#)
- value, [733](#)
- SimdChooser< T, true, true >, [733](#)
- value, [733](#)
- simdHalf
- Simd256\_impl< true, true, false, 2 >, [631](#)
- Simd256\_impl< true, true, false, 4 >, [643](#)
- Simd256\_impl< true, true, false, 8 >, [660](#)
- Simd256\_impl< true, true, true, 2 >, [669](#)
- Simd256\_impl< true, true, true, 4 >, [680](#)
- Simd256\_impl< true, true, true, 8 >, [696](#)

- Simd512\_impl< true, true, false, 8 >, [713](#)
  - Simd512\_impl< true, true, true, 8 >, [723](#)
- SimdSparseMatrix
  - FFLAS, [72](#)
- simdToType< T >, [733](#)
- Single, [733](#)
- size
  - BlockTransposeSIMD< Field, Simd, >, [413](#)
  - Info, [476](#), [477](#)
  - RNSInteger< RNS >, [541](#)
  - RNSIntegerMod< RNS >, [545](#)
- SIZEOF\_\_\_INT64\_T
  - config.h, [804](#)
- SIZEOF\_CHAR
  - config.h, [803](#)
- SIZEOF\_INT
  - config.h, [803](#)
- SIZEOF\_LONG
  - config.h, [804](#)
- SIZEOF\_LONG\_LONG
  - config.h, [804](#)
- SIZEOF\_SHORT
  - config.h, [804](#)
- sll
  - ScalFunctionsBase< Element, typename enable\_if< is\_integral< Element >::value >::type >, [560](#)
  - Simd128\_impl< true, true, false, 2 >, [568](#)
  - Simd128\_impl< true, true, false, 4 >, [578](#)
  - Simd128\_impl< true, true, false, 8 >, [588](#)
  - Simd128\_impl< true, true, true, 2 >, [596](#)
  - Simd128\_impl< true, true, true, 4 >, [605](#)
  - Simd128\_impl< true, true, true, 8 >, [614](#)
  - Simd256\_impl< true, true, false, 2 >, [635](#)
  - Simd256\_impl< true, true, false, 4 >, [650](#)
  - Simd256\_impl< true, true, false, 8 >, [664](#)
  - Simd256\_impl< true, true, true, 2 >, [671](#)
  - Simd256\_impl< true, true, true, 4 >, [682](#), [688](#)
  - Simd256\_impl< true, true, true, 8 >, [697](#)
  - Simd512\_impl< true, true, false, 8 >, [717](#)
  - Simd512\_impl< true, true, true, 8 >, [725](#)
- sll128
  - Simd128\_impl< true, true, false, 2 >, [571](#)
  - Simd128\_impl< true, true, false, 4 >, [581](#)
  - Simd128\_impl< true, true, false, 8 >, [592](#)
  - Simd128\_impl< true, true, true, 2 >, [601](#)
  - Simd128\_impl< true, true, true, 4 >, [609](#)
  - Simd128\_impl< true, true, true, 8 >, [619](#)
  - Simd128i\_base, [620](#)
- Solve
  - FFPACK, [333](#), [377](#)
- solve.C, [811](#)
  - main, [811](#)
- Solve\_modular\_double
  - ffpack.C, [998](#)
  - ffpack\_c.h, [1017](#)
- solveLB
  - FFPACK, [353](#), [377](#)
- solveLB2
  - FFPACK, [353](#), [377](#)
- solveLB2\_modular\_double
  - ffpack.C, [998](#)
  - ffpack\_c.h, [1018](#)
- solveLB\_modular\_double
  - ffpack.C, [998](#)
  - ffpack\_c.h, [1018](#)
- Sparse< \_Field, SparseMatrix\_t::COO >, [733](#)
  - col, [734](#)
  - dat, [734](#)
  - delayed, [734](#)
  - Field, [734](#)
  - kmax, [734](#)
  - m, [734](#)
  - maxrow, [735](#)
  - n, [734](#)
  - nElements, [735](#)
  - nnz, [734](#)
  - row, [734](#)
- Sparse< \_Field, SparseMatrix\_t::COO\_ZO >, [735](#)
  - col, [736](#)
  - cst, [736](#)
  - dat, [736](#)
  - delayed, [736](#)
  - Field, [735](#)
  - kmax, [736](#)
  - m, [736](#)
  - maxrow, [736](#)
  - n, [736](#)
  - nElements, [736](#)
  - nnz, [736](#)
  - row, [736](#)
- Sparse< \_Field, SparseMatrix\_t::CSR >, [737](#)
  - col, [738](#)
  - dat, [738](#)
  - delayed, [737](#)
  - Field, [737](#)
  - kmax, [737](#)
  - m, [737](#)
  - maxrow, [738](#)
  - n, [737](#)
  - nElements, [738](#)
  - nnz, [737](#)
  - st, [738](#)
  - stend, [738](#)
- Sparse< \_Field, SparseMatrix\_t::CSR\_HYB >, [738](#)
  - col, [739](#)
  - dat, [739](#)
  - delayed, [739](#)
  - Field, [739](#)
  - kmax, [739](#)
  - m, [739](#)
  - maxrow, [739](#)
  - n, [739](#)
  - nElements, [739](#)
  - nMOnes, [740](#)
  - nnz, [739](#)
  - nOnes, [740](#)

- nOthers, [740](#)
- st, [739](#)
- Sparse< \_Field, SparseMatrix\_t::CSR\_ZO >, [740](#)
  - col, [741](#)
  - cst, [741](#)
  - dat, [742](#)
  - delayed, [741](#)
  - Field, [741](#)
  - kmax, [741](#)
  - m, [741](#)
  - maxrow, [741](#)
  - n, [741](#)
  - nElements, [741](#)
  - nnz, [741](#)
  - st, [741](#)
  - stend, [741](#)
- Sparse< \_Field, SparseMatrix\_t::ELL >, [742](#)
  - col, [743](#)
  - dat, [743](#)
  - delayed, [742](#)
  - Field, [742](#)
  - kmax, [742](#)
  - ld, [743](#)
  - m, [743](#)
  - maxrow, [743](#)
  - n, [743](#)
  - nElements, [743](#)
  - nnz, [743](#)
- Sparse< \_Field, SparseMatrix\_t::ELL\_simd >, [743](#)
  - chunk, [744](#)
  - col, [745](#)
  - dat, [745](#)
  - delayed, [744](#)
  - kmax, [744](#)
  - ld, [744](#)
  - m, [744](#)
  - maxrow, [744](#)
  - n, [744](#)
  - nChunks, [745](#)
  - nElements, [744](#)
  - nnz, [744](#)
- Sparse< \_Field, SparseMatrix\_t::ELL\_simd\_ZO >, [745](#)
  - chunk, [746](#)
  - col, [746](#)
  - cst, [745](#)
  - dat, [746](#)
  - delayed, [745](#)
  - kmax, [746](#)
  - ld, [746](#)
  - m, [746](#)
  - maxrow, [746](#)
  - n, [746](#)
  - nChunks, [746](#)
  - nElements, [746](#)
  - nnz, [746](#)
- Sparse< \_Field, SparseMatrix\_t::ELL\_ZO >, [747](#)
  - col, [748](#)
  - cst, [747](#)
  - dat, [748](#)
  - delayed, [747](#)
  - Field, [747](#)
  - kmax, [747](#)
  - ld, [748](#)
  - m, [747](#)
  - maxrow, [748](#)
  - n, [747](#)
  - nElements, [748](#)
  - nnz, [748](#)
- Sparse< \_Field, SparseMatrix\_t::HYB\_ZO >, [748](#)
  - dat, [749](#)
  - delayed, [749](#)
  - Field, [749](#)
  - kmax, [749](#)
  - m, [749](#)
  - maxrow, [749](#)
  - mone, [749](#)
  - n, [749](#)
  - nElements, [749](#)
  - nnz, [749](#)
  - one, [749](#)
  - Self\_t, [749](#)
- Sparse< \_Field, SparseMatrix\_t::SELL >, [750](#)
  - chunk, [750](#)
  - chunkSize, [751](#)
  - col, [751](#)
  - dat, [752](#)
  - delayed, [750](#)
  - Field, [750](#)
  - kmax, [751](#)
  - m, [751](#)
  - maxrow, [751](#)
  - n, [751](#)
  - nChunks, [751](#)
  - nElements, [751](#)
  - nnz, [751](#)
  - perm, [751](#)
  - sigma, [751](#)
  - st, [751](#)
- Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, [752](#)
  - chunk, [753](#)
  - chunkSize, [754](#)
  - col, [754](#)
  - cst, [753](#)
  - dat, [754](#)
  - delayed, [753](#)
  - Field, [752](#)
  - kmax, [753](#)
  - m, [753](#)
  - maxrow, [753](#)
  - n, [753](#)
  - nChunks, [753](#)
  - nElements, [753](#)
  - nnz, [753](#)
  - perm, [753](#)
  - sigma, [753](#)
  - st, [753](#)

- Sparse< Field, SparseMatrix\_t, IdxT, PtrT >, 733
- sparse\_delete
  - FFLAS, 148–152, 155
- sparse\_init
  - FFLAS, 148–153, 155
- sparse\_matrix\_traits.h, 910
- sparse\_print
  - FFLAS, 149, 152, 155
- SparseMatrix\_t
  - FFLAS, 76
- SpecRankProfile
  - FFPACK, 376
- SpecRankProfile\_modular\_double
  - ffpack.C, 997
  - ffpack\_c.h, 1017
- splitt
  - parallel.h, 1036
- SPLITTER
  - parallel.h, 1036
- splitting\_0
  - parallel.h, 1036
- splitting\_1
  - parallel.h, 1036
- splitting\_2
  - parallel.h, 1036
- splitting\_3
  - parallel.h, 1036
- SpMat< Field, flag >, 754
  - \_coo, 754
  - \_csr, 754
  - \_ell, 754
- square\_inplace
  - FFLAS::\_ftranspose\_impl, 194
- sra
  - ScalFunctionsBase< Element, typename enable\_if<
    - is\_integral< Element >::value >::type >, 560
 Simd128\_impl< true, true, false, 2 >, 565
 Simd128\_impl< true, true, false, 4 >, 575
 Simd128\_impl< true, true, false, 8 >, 585
 Simd128\_impl< true, true, true, 2 >, 596
 Simd128\_impl< true, true, true, 4 >, 605
 Simd128\_impl< true, true, true, 8 >, 614
 Simd256\_impl< true, true, false, 2 >, 633
 Simd256\_impl< true, true, false, 4 >, 644, 647
 Simd256\_impl< true, true, false, 8 >, 661
 Simd256\_impl< true, true, true, 2 >, 671
 Simd256\_impl< true, true, true, 4 >, 682, 688
 Simd256\_impl< true, true, true, 8 >, 697
 Simd512\_impl< true, true, false, 8 >, 714
 Simd512\_impl< true, true, true, 8 >, 725
 >, 560
- srl
  - ScalFunctionsBase< Element, typename enable\_if<
    - is\_integral< Element >::value >::type >, 560
 Simd128\_impl< true, true, false, 2 >, 568
 Simd128\_impl< true, true, false, 4 >, 578
 Simd128\_impl< true, true, false, 8 >, 588
 Simd128\_impl< true, true, true, 2 >, 596
 Simd128\_impl< true, true, true, 4 >, 605
 >, 560
- Simd128\_impl< true, true, true, 8 >, 614
- Simd256\_impl< true, true, false, 2 >, 636
- Simd256\_impl< true, true, false, 4 >, 650
- Simd256\_impl< true, true, false, 8 >, 664
- Simd256\_impl< true, true, true, 2 >, 671
- Simd256\_impl< true, true, true, 4 >, 682, 688
- Simd256\_impl< true, true, true, 8 >, 697
- Simd512\_impl< true, true, false, 8 >, 717
- Simd512\_impl< true, true, true, 8 >, 725
- srl128
  - Simd128\_impl< true, true, false, 2 >, 571
  - Simd128\_impl< true, true, false, 4 >, 581
  - Simd128\_impl< true, true, false, 8 >, 592
  - Simd128\_impl< true, true, true, 2 >, 601
  - Simd128\_impl< true, true, true, 4 >, 610
  - Simd128\_impl< true, true, true, 8 >, 619
  - Simd128i\_base, 620
- sscal\_
  - config-blas.h, 821
- ST
  - fflas\_transpose.h, 913
- st
  - Sparse< \_Field, SparseMatrix\_t::CSR >, 738
  - Sparse< \_Field, SparseMatrix\_t::CSR\_HYB >, 739
  - Sparse< \_Field, SparseMatrix\_t::CSR\_ZO >, 741
  - Sparse< \_Field, SparseMatrix\_t::SELL >, 751
  - Sparse< \_Field, SparseMatrix\_t::SELL\_ZO >, 753
- StatsMatrix, 754
  - averageCol, 756
  - averageColDifference, 756
  - averageRow, 756
  - averageRowDifference, 756
  - coldim, 755
  - denseCols, 757
  - denseRows, 757
  - deviationCol, 756
  - deviationColDifference, 756
  - deviationRow, 756
  - deviationRowDifference, 756
  - maxCol, 756
  - maxColDifference, 756
  - maxRow, 755
  - maxRowDifference, 756
  - minCol, 756
  - minColDifference, 756
  - minRow, 756
  - minRowDifference, 756
  - nDenseCols, 757
  - nDenseRows, 757
  - nEmptyCols, 757
  - nEmptyColsEnd, 757
  - nEmptyRows, 757
  - nMOnes, 755
  - nnz, 755
  - nOnes, 755
  - nOthers, 755
  - rowdim, 755



- STD\_RECINT\_SIZE
  - benchmark-fgemm-mp.C, [784](#)
  - benchmark-fgemv-mp.C, [787](#)
- STDC\_HEADERS
  - config.h, [804](#)
- stend
  - Sparse< \_Field, SparseMatrix\_t::CSR >, [738](#)
  - Sparse< \_Field, SparseMatrix\_t::CSR\_ZO >, [741](#)
- store
  - Simd128\_impl< true, true, false, 2 >, [565](#), [568](#)
  - Simd128\_impl< true, true, false, 4 >, [575](#), [578](#)
  - Simd128\_impl< true, true, false, 8 >, [585](#), [588](#)
  - Simd128\_impl< true, true, true, 2 >, [596](#)
  - Simd128\_impl< true, true, true, 4 >, [604](#)
  - Simd128\_impl< true, true, true, 8 >, [614](#)
  - Simd256\_impl< true, false, true, 8 >, [624](#)
  - Simd256\_impl< true, true, false, 2 >, [632](#), [635](#)
  - Simd256\_impl< true, true, false, 4 >, [644](#), [647](#), [650](#)
  - Simd256\_impl< true, true, false, 8 >, [661](#), [663](#)
  - Simd256\_impl< true, true, true, 2 >, [671](#)
  - Simd256\_impl< true, true, true, 4 >, [682](#), [688](#)
  - Simd256\_impl< true, true, true, 8 >, [697](#)
  - Simd512\_impl< true, false, true, 8 >, [706](#)
  - Simd512\_impl< true, true, false, 8 >, [714](#), [717](#)
  - Simd512\_impl< true, true, true, 8 >, [725](#)
- storeu
  - Simd128\_impl< true, true, false, 2 >, [565](#), [568](#)
  - Simd128\_impl< true, true, false, 4 >, [575](#), [578](#)
  - Simd128\_impl< true, true, false, 8 >, [585](#), [588](#)
  - Simd128\_impl< true, true, true, 2 >, [596](#)
  - Simd128\_impl< true, true, true, 4 >, [605](#)
  - Simd128\_impl< true, true, true, 8 >, [614](#)
  - Simd256\_impl< true, false, true, 8 >, [624](#)
  - Simd256\_impl< true, true, false, 2 >, [633](#), [635](#)
  - Simd256\_impl< true, true, false, 4 >, [644](#), [647](#), [650](#)
  - Simd256\_impl< true, true, false, 8 >, [661](#), [664](#)
  - Simd256\_impl< true, true, true, 2 >, [671](#)
  - Simd256\_impl< true, true, true, 4 >, [682](#), [688](#)
  - Simd256\_impl< true, true, true, 8 >, [697](#)
  - Simd512\_impl< true, false, true, 8 >, [706](#)
  - Simd512\_impl< true, true, false, 8 >, [714](#), [717](#)
  - Simd512\_impl< true, true, true, 8 >, [725](#)
- stream
  - Simd128\_impl< true, true, false, 2 >, [565](#), [568](#)
  - Simd128\_impl< true, true, false, 4 >, [575](#), [578](#)
  - Simd128\_impl< true, true, false, 8 >, [585](#), [588](#)
  - Simd128\_impl< true, true, true, 2 >, [596](#)
  - Simd128\_impl< true, true, true, 4 >, [605](#)
  - Simd128\_impl< true, true, true, 8 >, [614](#)
  - Simd256\_impl< true, false, true, 8 >, [624](#)
  - Simd256\_impl< true, true, false, 2 >, [633](#), [635](#)
  - Simd256\_impl< true, true, false, 4 >, [644](#), [647](#), [650](#)
  - Simd256\_impl< true, true, false, 8 >, [661](#), [664](#)
  - Simd256\_impl< true, true, true, 2 >, [671](#)
  - Simd256\_impl< true, true, true, 4 >, [682](#), [688](#)
- Simd256\_impl< true, true, true, 8 >, [697](#)
- Simd512\_impl< true, false, true, 8 >, [706](#)
- Simd512\_impl< true, true, false, 8 >, [714](#), [717](#)
- Simd512\_impl< true, true, true, 8 >, [725](#)
- strmm\_
  - config-blas.h, [821](#)
- strsm\_
  - config-blas.h, [821](#)
- sub
  - FFLAS::vectorised, [285](#)
  - FieldSimd< \_Field >, [453](#)
  - RNSIntegerMod< RNS >, [547](#)
  - ScalFunctions< Element >, [553](#)
  - Simd128\_impl< true, true, false, 2 >, [570](#)
  - Simd128\_impl< true, true, false, 4 >, [580](#)
  - Simd128\_impl< true, true, false, 8 >, [590](#)
  - Simd128\_impl< true, true, true, 2 >, [598](#)
  - Simd128\_impl< true, true, true, 4 >, [606](#)
  - Simd128\_impl< true, true, true, 8 >, [616](#)
  - Simd256\_impl< true, false, true, 8 >, [626](#)
  - Simd256\_impl< true, true, false, 2 >, [637](#)
  - Simd256\_impl< true, true, false, 4 >, [654](#)
  - Simd256\_impl< true, true, false, 8 >, [665](#)
  - Simd256\_impl< true, true, true, 2 >, [673](#)
  - Simd256\_impl< true, true, true, 4 >, [684](#), [690](#)
  - Simd256\_impl< true, true, true, 8 >, [699](#)
  - Simd512\_impl< true, false, true, 8 >, [708](#)
  - Simd512\_impl< true, true, false, 8 >, [719](#)
  - Simd512\_impl< true, true, true, 8 >, [727](#)
- sub\_r
  - FieldSimd< \_Field >, [453](#)
- subin
  - FieldSimd< \_Field >, [453](#)
  - ScalFunctions< Element >, [553](#)
  - Simd128\_impl< true, true, false, 2 >, [570](#)
  - Simd128\_impl< true, true, false, 4 >, [580](#)
  - Simd128\_impl< true, true, false, 8 >, [590](#)
  - Simd128\_impl< true, true, true, 2 >, [598](#)
  - Simd128\_impl< true, true, true, 4 >, [607](#)
  - Simd128\_impl< true, true, true, 8 >, [616](#)
  - Simd256\_impl< true, false, true, 8 >, [626](#)
  - Simd256\_impl< true, true, false, 2 >, [637](#)
  - Simd256\_impl< true, true, false, 4 >, [654](#)
  - Simd256\_impl< true, true, false, 8 >, [665](#)
  - Simd256\_impl< true, true, true, 2 >, [673](#)
  - Simd256\_impl< true, true, true, 4 >, [684](#), [690](#)
  - Simd256\_impl< true, true, true, 8 >, [699](#)
  - Simd512\_impl< true, false, true, 8 >, [708](#)
  - Simd512\_impl< true, true, false, 8 >, [719](#)
  - Simd512\_impl< true, true, true, 8 >, [727](#)
- subin\_r
  - FieldSimd< \_Field >, [453](#)
- subp
  - FFLAS::vectorised, [285](#)
- support\_fast\_mod< double >, [757](#)
- support\_fast\_mod< float >, [758](#)
- support\_fast\_mod< int64\_t >, [758](#)
- support\_fast\_mod< T >, [757](#)

- support\_simd< T >, 758
- support\_simd\_add< T >, 759
- support\_simd\_mod< T >, 759
- swapval
  - FFPACK, 389
- SYNCH\_GROUP
  - parallel.h, 1032
- SysTimer
  - FFLAS, 74
- T
  - limits< char >, 486
  - limits< double >, 486
  - limits< float >, 487
  - limits< Givaro::Integer >, 488
  - limits< int >, 488
  - limits< long >, 489
  - limits< long long >, 489
  - limits< Reclnt::rint< K > >, 490
  - limits< Reclnt::ruint< K > >, 491
  - limits< short int >, 491
  - limits< signed char >, 492
  - limits< unsigned char >, 492
  - limits< unsigned int >, 493
  - limits< unsigned long >, 494
  - limits< unsigned long long >, 494
  - limits< unsigned short int >, 495
- t
  - Simd128\_impl< true, true, false, 2 >::Converter, 429
  - Simd128\_impl< true, true, false, 4 >::Converter, 429
  - Simd128\_impl< true, true, false, 8 >::Converter, 429
  - Simd128\_impl< true, true, true, 2 >::Converter, 430
  - Simd128\_impl< true, true, true, 4 >::Converter, 430
  - Simd128\_impl< true, true, true, 8 >::Converter, 430
  - Simd256\_impl< true, false, true, 8 >::Converter, 431
  - Simd256\_impl< true, true, false, 2 >::Converter, 431
  - Simd256\_impl< true, true, false, 4 >::Converter, 431
  - Simd256\_impl< true, true, false, 8 >::Converter, 432
  - Simd256\_impl< true, true, true, 2 >::Converter, 432
  - Simd256\_impl< true, true, true, 4 >::Converter, 432
  - Simd256\_impl< true, true, true, 8 >::Converter, 433
  - Simd512\_impl< true, true, false, 8 >::Converter, 433
  - Simd512\_impl< true, true, true, 8 >::Converter, 433
- TASK
  - parallel.h, 1031
- tBC
  - test-lu.C, 1089
  - test-permutations.C, 1094
- Test
  - Test< Elt >, 761
- test
  - test-maxdelayeddim.C, 1090
- Test< Elt >, 759
  - \_mm, 762
  - \_nn, 762
  - cardinality, 761
  - doTests, 761
  - Elt\_ptr, 760
  - enable\_if\_no\_simd\_t, 760
  - enable\_if\_simd128\_t, 760
  - enable\_if\_simd256\_t, 760
  - enable\_if\_simd512\_t, 761
  - enable\_if\_t, 760
  - F, 762
  - Field, 760
  - is\_same\_element, 760
  - Residu, 760
  - run, 761
  - Test, 761
  - test\_ftranspose, 761
- test-charpoly-check.C, 1052
  - ENABLE\_CHECKER\_charpoly, 1052
  - main, 1052
  - printPolynomial, 1052
  - TIME\_CHECKER\_CHARPOLY, 1052
- test-charpoly.C, 1052
  - launch\_test, 1053
  - main, 1053
  - run\_with\_field, 1053
- test-compressQ.C, 1053
  - Field, 1054
  - main, 1054
  - printvect, 1054
- test-det-check.C, 1054
  - ENABLE\_CHECKER\_Det, 1054
  - main, 1055
  - TIME\_CHECKER\_Det, 1054
- test-det.C, 1055
  - main, 1055
  - test\_det, 1055
- test-echelon.C, 1055
  - \_\_FFLASFFPACK\_GAUSSJORDAN\_BASECASE, 1056
  - \_\_FFLASFFPACK\_PLUQ\_THRESHOLD, 1056
  - \_\_FFLASFFPACK\_SEQUENTIAL, 1056
  - main, 1057
  - run\_with\_field, 1057
  - test\_colechelon, 1056
  - test\_redcoechelon, 1057
  - test\_redrowechelon, 1057
  - test\_rowechelon, 1057
- test-fadd.C, 1058

- main, 1059
- test\_fadd, 1058
- test\_faddin, 1058
- test\_fsub, 1058
- test\_fsubin, 1059
- test-fdot.C, 1059
  - check\_fdot, 1060
  - ENABLE\_ALL\_CHECKINGS, 1059
  - main, 1060
  - run\_with\_field, 1060
  - run\_with\_Integer, 1060
- test-fgemm-check.C, 1060
  - ENABLE\_ALL\_CHECKINGS, 1061
  - launch\_MM\_dispatch, 1061
  - main, 1061
  - run\_with\_field, 1061
- test-fgemm.C, 1061
  - check\_MM, 1062
  - ENABLE\_CHECKER\_fgemm, 1062
  - launch\_MM, 1062
  - launch\_MM\_dispatch, 1063
  - main, 1063
  - run\_with\_field, 1063
- test-fgemv.C, 1064
  - check\_MV, 1064
  - launch\_MV, 1064
  - launch\_MV\_dispatch, 1065
  - main, 1065
  - run\_with\_field, 1065
- test-fger.C, 1065
  - check\_fger, 1066
  - launch\_fger, 1066
  - launch\_fger\_dispatch, 1066
  - main, 1067
  - run\_with\_field, 1067
  - TIME, 1066
- test-fgesv.C, 1067
  - main, 1068
  - run\_with\_field, 1068
  - test\_rect\_fgesv, 1068
  - test\_square\_fgesv, 1068
- test-finit.C, 1068
  - main, 1069
  - run\_with\_field, 1069
  - test\_freduce, 1069
- test-fscal.C, 1069
  - main, 1071
  - test\_fscal, 1070
  - test\_fscaln, 1070
- test-fsyr2k.C, 1071
  - check\_fsyr2k, 1071
  - ENABLE\_ALL\_CHECKINGS, 1071
  - main, 1072
  - run\_with\_field, 1071
- test-fsyrk.C, 1072
  - check\_computeS1S2, 1073
  - check\_fsyrk, 1073
  - check\_fsyrk\_bkdiag, 1073
  - check\_fsyrk\_diag, 1073
  - ENABLE\_ALL\_CHECKINGS, 1073
  - main, 1074
  - run\_with\_field, 1073
- test-fsytrf.C, 1074
  - main, 1075
  - operator<<, 1074
  - run\_with\_field, 1075
  - test\_generic\_fsytrf, 1075
  - test\_RPM\_fsytrf, 1074
- test-ftrmm.C, 1075
  - \_\_FFLASFFPACK\_SEQUENTIAL, 1076
  - check\_ftrmm, 1076
  - main, 1076
  - run\_with\_field, 1076
- test-ftrmv.C, 1076
  - \_\_FFLASFFPACK\_SEQUENTIAL, 1077
  - check\_ftrmv, 1077
  - ENABLE\_ALL\_CHECKINGS, 1077
  - main, 1077
  - run\_with\_field, 1077
- test-ftrsm-check.C, 1078
  - ENABLE\_ALL\_CHECKINGS, 1078
  - main, 1078
- test-ftrsm.C, 1078
  - \_\_FFLASFFPACK\_SEQUENTIAL, 1079
  - check\_ftrsm, 1079
  - ENABLE\_ALL\_CHECKINGS, 1079
  - main, 1079
  - run\_with\_field, 1079
- test-ftrssyr2k.C, 1079
  - check\_ftrssyr2k, 1080
  - ENABLE\_ALL\_CHECKINGS, 1080
  - main, 1080
  - run\_with\_field, 1080
- test-ftrstr.C, 1081
  - check\_ftrstr, 1081
  - ENABLE\_ALL\_CHECKINGS, 1081
  - main, 1081
  - run\_with\_field, 1081
- test-ftrsv.C, 1082
  - \_\_FFLASFFPACK\_SEQUENTIAL, 1082
  - check\_ftrsv, 1082
  - ENABLE\_ALL\_CHECKINGS, 1082
  - main, 1083
  - run\_with\_field, 1082
- test-ftrtri.C, 1083
  - \_\_FFLASFFPACK\_SEQUENTIAL, 1083
  - check\_ftrtri, 1083
  - ENABLE\_ALL\_CHECKINGS, 1083
  - main, 1084
  - run\_with\_field, 1084
- test-interfaces-c.c, 1084
  - main, 1084
- test-invert-check.C, 1084
  - ENABLE\_ALL\_CHECKINGS, 1085
  - main, 1085
- test-io.C, 1085

- main, 1085
- run\_with\_field, 1085
- test-lu.C, 1086
  - \_\_FFLASFFPACK\_SEQUENTIAL, 1086
  - \_\_LUDIVINE\_CUTOFF, 1087
  - BASECASE\_K, 1086
  - launch\_test, 1088
  - main, 1089
  - mvcnt, 1089
  - run\_with\_field, 1089
  - tBC, 1089
  - test\_LUdivine, 1087
  - test\_pluq, 1088
  - tgemm, 1089
  - timtot, 1089
  - tperm, 1089
  - trest, 1089
  - ttrsm, 1089
  - verifPLUQ, 1087
- test-maxdelayeddim.C, 1090
  - main, 1090
  - MAX\_WITH\_SIZE\_T, 1090
  - test, 1090
- test-minpoly.C, 1090
  - check\_minpoly, 1091
  - main, 1091
  - run\_with\_field, 1091
- test-multifile1.C, 1091
- test-multifile2.C, 1091
  - main, 1091
- test-nullspace.C, 1092
  - checkingMessage, 1092
  - main, 1093
  - readOrRandomMatrixWithRankAndRandomRPM, 1092
  - run\_with\_field, 1093
  - test\_nullspace, 1092
- test-permutations.C, 1093
  - checkMonotonicApplyP, 1093
  - main, 1093
  - tBC, 1094
  - tgemm, 1094
  - timtot, 1094
  - tperm, 1094
  - trest, 1094
  - ttrsm, 1094
- test-pluq-check.C, 1094
  - ENABLE\_ALL\_CHECKINGS, 1094
  - main, 1094
- test-quasisep.C, 1095
  - launch\_test, 1095
  - main, 1096
  - run\_with\_field, 1096
  - test\_BruhatGenerator, 1095
  - testLTQSRPM, 1095
- test-rankprofiles.C, 1096
  - \_\_FFLASFFPACK\_SEQUENTIAL, 1096
  - main, 1097
  - run\_with\_field, 1096
- test-rpm.C, 1097
  - checkRPM, 1097
  - checkSymmetricRPM, 1097
  - main, 1097
- test-simd.C, 1097
  - \_TEST\_ONE, 1099
  - check\_eq, 1099, 1100
  - cmp, 1100
  - eval\_func\_on\_array, 1100
  - main, 1100
  - operator<<, 1100
  - TEST\_IMPL, 1099
  - test\_impl, 1100
  - test\_impl\_base, 1100
  - TEST\_ONE\_OP, 1099
  - TEST\_ONE\_OP\_WZ, 1099
- test-solve.C, 1101
  - check\_solve, 1101
  - main, 1101
  - run\_with\_field, 1101
- test-storage-transpose.C, 1101
  - main, 1102
- test-utils.h, 1047
- test\_BruhatGenerator
  - test-quasisep.C, 1095
- test\_colechelon
  - test-echelon.C, 1056
- test\_det
  - test-det.C, 1055
- test\_fadd
  - test-fadd.C, 1058
- test\_faddin
  - test-fadd.C, 1058
- test\_freduce
  - test-finit.C, 1069
- test\_fscal
  - test-fscal.C, 1070
- test\_fscalin
  - test-fscal.C, 1070
- test\_fsub
  - test-fadd.C, 1058
- test\_fsubin
  - test-fadd.C, 1059
- test\_ftranspose
  - Test< Elt >, 761
- test\_generic\_fsytrf
  - test-fsytrf.C, 1075
- TEST\_IMPL
  - test-simd.C, 1099
- test\_impl
  - test-simd.C, 1100
- test\_impl\_base
  - test-simd.C, 1100
- test\_LUdivine
  - test-lu.C, 1087
- test\_nullspace
  - test-nullspace.C, 1092

- TEST\_ONE\_OP
  - test-simd.C, [1099](#)
- TEST\_ONE\_OP\_WZ
  - test-simd.C, [1099](#)
- test\_pluq
  - test-lu.C, [1088](#)
- test\_rect\_fgesv
  - test-fgesv.C, [1068](#)
- test\_redcochelon
  - test-echelon.C, [1057](#)
- test\_redrowechelon
  - test-echelon.C, [1057](#)
- test\_rowechelon
  - test-echelon.C, [1057](#)
- test\_RPM\_fsytrf
  - test-fsytrf.C, [1074](#)
- test\_square\_fgesv
  - test-fgesv.C, [1068](#)
- testLTQSRPM
  - test-quasisep.C, [1095](#)
- TestOneMethod
  - TestOneMethod< Simd >, [763](#)
- TestOneMethod< Simd >, [762](#)
  - Element, [763](#)
  - enable\_if\_t, [763](#)
  - evaluate\_scalar\_method, [763](#), [764](#)
  - evaluate\_simd\_method, [764](#)
  - getStatus, [764](#)
  - getTestName, [764](#)
  - inputs, [765](#)
  - name, [765](#)
  - nb\_lref, [764](#)
  - outputs\_scalar, [765](#)
  - outputs\_simd, [765](#)
  - TestOneMethod, [763](#)
  - vect\_size, [764](#)
  - vect\_t, [763](#)
  - vectElt, [763](#)
  - writeDebugData, [764](#)
  - writeResultLine, [764](#)
- tfn\_minus, [765](#)
  - operator(), [765](#)
- tfn\_minus\_eq, [765](#)
  - operator(), [766](#)
- tfn\_mul, [766](#)
  - operator(), [766](#)
- tfn\_mul\_eq, [766](#)
  - operator(), [766](#)
- tfn\_plus, [766](#)
  - operator(), [767](#)
- tfn\_plus\_eq, [767](#)
  - operator(), [767](#)
- tgemm
  - test-lu.C, [1089](#)
  - test-permutations.C, [1094](#)
- THREAD\_INDEX
  - parallel.h, [1032](#)
- THREADS
  - benchmark-fgemm-rns.C, [785](#)
- Threads, [767](#)
- threads\_fgemm
  - FFPACK, [368](#)
- threads\_ftsm
  - FFPACK, [368](#)
- THREED
  - benchmark-fgemm-rns.C, [785](#)
- ThreeD, [767](#)
- THREEDA
  - benchmark-fgemm-rns.C, [786](#)
- ThreeDAdaptive, [767](#)
- ThreeDInPlace, [768](#)
- THREEDIP
  - benchmark-fgemm-rns.C, [786](#)
- TIME
  - test-fger.C, [1066](#)
- TIME\_CHECKER\_CHARPOLY
  - test-charpoly-check.C, [1052](#)
- TIME\_CHECKER\_Det
  - test-det-check.C, [1054](#)
- Timer
  - FFLAS, [74](#)
- timer.h, [1048](#)
- timtot
  - test-lu.C, [1089](#)
  - test-permutations.C, [1094](#)
- TInverter
  - FFPACK, [351](#), [353](#)
- tmain
  - benchmark-fgemm-mp.C, [784](#)
  - benchmark-fgemv-mp.C, [787](#)
- Todo List, [9](#)
- tperm
  - test-lu.C, [1089](#)
  - test-permutations.C, [1094](#)
- transpose
  - BlockTransposeSIMD< Field, Simd, >, [414](#)
  - Simd128\_impl< true, true, false, 2 >, [569](#)
  - Simd128\_impl< true, true, false, 4 >, [579](#)
  - Simd128\_impl< true, true, false, 8 >, [589](#)
  - Simd128\_impl< true, true, true, 2 >, [597](#)
  - Simd128\_impl< true, true, true, 4 >, [606](#)
  - Simd128\_impl< true, true, true, 8 >, [615](#)
  - Simd256\_impl< true, false, true, 8 >, [625](#)
  - Simd256\_impl< true, true, false, 2 >, [637](#)
  - Simd256\_impl< true, true, false, 4 >, [653](#)
  - Simd256\_impl< true, true, false, 8 >, [665](#)
  - Simd256\_impl< true, true, true, 2 >, [672](#)
  - Simd256\_impl< true, true, true, 4 >, [683](#), [689](#)
  - Simd256\_impl< true, true, true, 8 >, [699](#)
  - Simd512\_impl< true, false, true, 8 >, [707](#)
  - Simd512\_impl< true, true, false, 8 >, [718](#)
  - Simd512\_impl< true, true, true, 8 >, [726](#)
- trest
  - test-lu.C, [1089](#)
  - test-permutations.C, [1094](#)
- trinv\_left

- FFPACK, [314](#), [372](#)
- trinv\_left\_modular\_double
  - ffpack.C, [990](#)
  - ffpack\_c.h, [1011](#)
- TRSMBound
  - FFLAS::Protected, [216](#), [217](#)
- TRSMHelper
  - TRSMHelper< ReclterTrait, ParSeqTrait >, [768](#)
- TRSMHelper< ReclterTrait, ParSeqTrait >, [768](#)
  - parseq, [769](#)
  - pMMH, [769](#)
  - TRSMHelper, [768](#)
- TTimer
  - arithprog.C, [771](#)
  - benchmark-dgemm.C, [778](#)
  - benchmark-dgetrf.C, [779](#)
  - benchmark-dgetri.C, [780](#)
  - benchmark-dsytrf.C, [780](#)
  - benchmark-dtrsm.C, [781](#)
  - benchmark-dtrtri.C, [782](#)
  - charpoly.C, [808](#)
  - fsyrk.C, [772](#)
  - fsytrf.C, [773](#)
  - fttrtri.C, [774](#)
  - pluq.C, [810](#)
  - winograd.C, [775](#)
- ttrsm
  - test-lu.C, [1089](#)
  - test-permutations.C, [1094](#)
- Tutorial, [2](#)
- TWOD
  - benchmark-fgemm-rns.C, [785](#)
- TwoD, [769](#)
- TWODA
  - benchmark-fgemm-rns.C, [785](#)
- TwoDAdaptive, [769](#)
- type
  - Argument, [408](#)
  - associatedDelayedField< const FFPACK::RNSIntegerMod>
    - RNS > >, [408](#)
  - associatedDelayedField< const Givaro::Modular<
    - T, X > >, [409](#)
  - associatedDelayedField< const Givaro::ModularBalanced>
    - T > >, [409](#)
  - associatedDelayedField< const Givaro::ZRing< T
    - > >, [410](#)
  - associatedDelayedField< Field >, [408](#)
  - CompactElement< double >, [425](#)
  - CompactElement< Element >, [425](#)
  - CompactElement< float >, [426](#)
  - CompactElement< int16\_t >, [426](#)
  - CompactElement< int32\_t >, [426](#)
  - CompactElement< int64\_t >, [426](#)
  - is\_simd< T >, [479](#)
- TYPE\_BOOL
  - args-parser.h, [1039](#)
- TYPE\_DOUBLE
  - args-parser.h, [1039](#)
- TYPE\_INT
  - args-parser.h, [1039](#)
- TYPE\_INTEGER
  - args-parser.h, [1039](#)
- type\_integer
  - args-parser.h, [1039](#)
- TYPE\_INTLIST
  - args-parser.h, [1039](#)
- TYPE\_LONGLONG
  - args-parser.h, [1039](#)
- TYPE\_NONE
  - args-parser.h, [1039](#)
- TYPE\_STR
  - args-parser.h, [1039](#)
- type\_string
  - NoSimd< T >, [519](#)
  - Simd128\_impl< true, true, false, 2 >, [564](#)
  - Simd128\_impl< true, true, false, 4 >, [575](#)
  - Simd128\_impl< true, true, false, 8 >, [585](#)
  - Simd128\_impl< true, true, true, 2 >, [595](#)
  - Simd128\_impl< true, true, true, 4 >, [604](#)
  - Simd128\_impl< true, true, true, 8 >, [613](#)
  - Simd256\_impl< true, false, true, 8 >, [623](#)
  - Simd256\_impl< true, true, false, 2 >, [632](#)
  - Simd256\_impl< true, true, false, 4 >, [644](#), [646](#)
  - Simd256\_impl< true, true, false, 8 >, [660](#)
  - Simd256\_impl< true, true, true, 2 >, [670](#)
  - Simd256\_impl< true, true, true, 4 >, [681](#), [687](#)
  - Simd256\_impl< true, true, true, 8 >, [696](#)
  - Simd512\_impl< true, false, true, 8 >, [705](#)
  - Simd512\_impl< true, true, false, 8 >, [713](#)
  - Simd512\_impl< true, true, true, 8 >, [724](#)
- TYPE\_UINT64
  - args-parser.h, [1039](#)
- unfit
  - FFLAS::Protected, [225](#), [226](#)
- unpackhi
  - ScalFunctions< Element >, [555](#)
  - Simd128\_impl< true, true, false, 2 >, [568](#)
  - Simd128\_impl< true, true, false, 4 >, [579](#)
  - Simd128\_impl< true, true, false, 8 >, [589](#)
  - Simd128\_impl< true, true, true, 2 >, [597](#)
  - Simd128\_impl< true, true, true, 4 >, [605](#)
  - Simd128\_impl< true, true, true, 8 >, [615](#)
  - Simd256\_impl< true, false, true, 8 >, [625](#)
  - Simd256\_impl< true, true, false, 2 >, [636](#)
  - Simd256\_impl< true, true, false, 4 >, [651](#), [652](#)
  - Simd256\_impl< true, true, false, 8 >, [664](#)
  - Simd256\_impl< true, true, true, 2 >, [672](#)
  - Simd256\_impl< true, true, true, 4 >, [683](#), [689](#)
  - Simd256\_impl< true, true, true, 8 >, [698](#)
  - Simd512\_impl< true, false, true, 8 >, [707](#)
  - Simd512\_impl< true, true, false, 8 >, [718](#)
  - Simd512\_impl< true, true, true, 8 >, [726](#)
- unpackhi\_intrinsic
  - Simd128\_impl< true, true, false, 2 >, [568](#)
  - Simd128\_impl< true, true, false, 4 >, [578](#)
  - Simd128\_impl< true, true, false, 8 >, [589](#)



- Simd128\_impl< true, true, true, 2 >, [596](#)
- Simd128\_impl< true, true, true, 4 >, [605](#)
- Simd128\_impl< true, true, true, 8 >, [614](#)
- Simd256\_impl< true, false, true, 8 >, [625](#)
- Simd256\_impl< true, true, false, 2 >, [636](#)
- Simd256\_impl< true, true, false, 4 >, [651](#)
- Simd256\_impl< true, true, false, 8 >, [664](#)
- Simd256\_impl< true, true, true, 2 >, [672](#)
- Simd256\_impl< true, true, true, 4 >, [682](#), [689](#)
- Simd256\_impl< true, true, true, 8 >, [698](#)
- Simd512\_impl< true, false, true, 8 >, [707](#)
- Simd512\_impl< true, true, false, 8 >, [718](#)
- Simd512\_impl< true, true, true, 8 >, [726](#)
- unpacklo
  - ScalFunctions< Element >, [555](#)
  - Simd128\_impl< true, true, false, 2 >, [568](#)
  - Simd128\_impl< true, true, false, 4 >, [578](#)
  - Simd128\_impl< true, true, false, 8 >, [589](#)
  - Simd128\_impl< true, true, true, 2 >, [596](#)
  - Simd128\_impl< true, true, true, 4 >, [605](#)
  - Simd128\_impl< true, true, true, 8 >, [615](#)
  - Simd256\_impl< true, false, true, 8 >, [625](#)
  - Simd256\_impl< true, true, false, 2 >, [636](#)
  - Simd256\_impl< true, true, false, 4 >, [651](#)
  - Simd256\_impl< true, true, false, 8 >, [664](#)
  - Simd256\_impl< true, true, true, 2 >, [672](#)
  - Simd256\_impl< true, true, true, 4 >, [683](#), [689](#)
  - Simd256\_impl< true, true, true, 8 >, [698](#)
  - Simd512\_impl< true, false, true, 8 >, [707](#)
  - Simd512\_impl< true, true, false, 8 >, [718](#)
  - Simd512\_impl< true, true, true, 8 >, [726](#)
- unpacklo\_intrinsic
  - Simd128\_impl< true, true, false, 2 >, [568](#)
  - Simd128\_impl< true, true, false, 4 >, [578](#)
  - Simd128\_impl< true, true, false, 8 >, [588](#)
  - Simd128\_impl< true, true, true, 2 >, [596](#)
  - Simd128\_impl< true, true, true, 4 >, [605](#)
  - Simd128\_impl< true, true, true, 8 >, [614](#)
  - Simd256\_impl< true, false, true, 8 >, [625](#)
  - Simd256\_impl< true, true, false, 2 >, [636](#)
  - Simd256\_impl< true, true, false, 4 >, [651](#)
  - Simd256\_impl< true, true, false, 8 >, [664](#)
  - Simd256\_impl< true, true, true, 2 >, [671](#)
  - Simd256\_impl< true, true, true, 4 >, [682](#), [688](#)
  - Simd256\_impl< true, true, true, 8 >, [698](#)
  - Simd512\_impl< true, false, true, 8 >, [706](#)
  - Simd512\_impl< true, true, false, 8 >, [717](#)
  - Simd512\_impl< true, true, true, 8 >, [726](#)
- unpacklohi
  - ScalFunctions< Element >, [555](#)
  - Simd128\_impl< true, true, false, 2 >, [569](#)
  - Simd128\_impl< true, true, false, 4 >, [579](#)
  - Simd128\_impl< true, true, false, 8 >, [589](#)
  - Simd128\_impl< true, true, true, 2 >, [597](#)
  - Simd128\_impl< true, true, true, 4 >, [606](#)
  - Simd128\_impl< true, true, true, 8 >, [615](#)
  - Simd256\_impl< true, false, true, 8 >, [625](#)
  - Simd256\_impl< true, true, false, 2 >, [636](#)
- Simd256\_impl< true, true, false, 4 >, [652](#)
- Simd256\_impl< true, true, false, 8 >, [664](#)
- Simd256\_impl< true, true, true, 2 >, [672](#)
- Simd256\_impl< true, true, true, 4 >, [683](#), [689](#)
- Simd256\_impl< true, true, true, 8 >, [698](#)
- Simd512\_impl< true, false, true, 8 >, [707](#)
- Simd512\_impl< true, true, false, 8 >, [718](#)
- Simd512\_impl< true, true, true, 8 >, [726](#)
- UnparametricTag, [769](#)
- updateD
  - FFPACK::Protected, [401](#)
- USE\_OPENMP
  - config.h, [804](#)
- UserTimer
  - FFLAS, [74](#)
- utils.h, [912](#)
- v
  - Simd128\_impl< true, true, false, 2 >::Converter, [429](#)
  - Simd128\_impl< true, true, false, 4 >::Converter, [429](#)
  - Simd128\_impl< true, true, false, 8 >::Converter, [429](#)
  - Simd128\_impl< true, true, true, 2 >::Converter, [430](#)
  - Simd128\_impl< true, true, true, 4 >::Converter, [430](#)
  - Simd128\_impl< true, true, true, 8 >::Converter, [430](#)
  - Simd256\_impl< true, false, true, 8 >::Converter, [431](#)
  - Simd256\_impl< true, true, false, 2 >::Converter, [431](#)
  - Simd256\_impl< true, true, false, 4 >::Converter, [431](#)
  - Simd256\_impl< true, true, false, 8 >::Converter, [432](#)
  - Simd256\_impl< true, true, true, 2 >::Converter, [432](#)
  - Simd256\_impl< true, true, true, 4 >::Converter, [432](#)
  - Simd256\_impl< true, true, true, 8 >::Converter, [433](#)
  - Simd512\_impl< true, true, false, 8 >::Converter, [433](#)
  - Simd512\_impl< true, true, true, 8 >::Converter, [433](#)
- val
  - Coo< Field >, [436](#)
  - Coo< ValT, IdxT >, [435](#), [438](#)
- valid
  - NoSimd< T >, [519](#)
  - Simd128\_impl< true, true, false, 2 >, [567](#)
  - Simd128\_impl< true, true, false, 4 >, [577](#)
  - Simd128\_impl< true, true, false, 8 >, [587](#)
  - Simd128\_impl< true, true, true, 2 >, [595](#)
  - Simd128\_impl< true, true, true, 4 >, [604](#)
  - Simd128\_impl< true, true, true, 8 >, [613](#)

- Simd256\_impl< true, false, true, 8 >, [623](#)
- Simd256\_impl< true, true, false, 2 >, [634](#)
- Simd256\_impl< true, true, false, 4 >, [648](#), [649](#)
- Simd256\_impl< true, true, false, 8 >, [663](#)
- Simd256\_impl< true, true, true, 2 >, [670](#)
- Simd256\_impl< true, true, true, 4 >, [681](#), [687](#)
- Simd256\_impl< true, true, true, 8 >, [696](#)
- Simd512\_impl< true, false, true, 8 >, [705](#)
- Simd512\_impl< true, true, false, 8 >, [716](#)
- Simd512\_impl< true, true, true, 8 >, [724](#)
- VALUE
  - parallel.h, [1032](#)
- value
  - AlgoChooser< ModeCategories::ConvertTo< ElementCategories::RNSElementTag >, ParSeq >, [405](#)
  - AlgoChooser< ModeT, ParSeq >, [405](#)
  - ALL< false, v... >, [406](#)
  - ALL< true, v... >, [406](#)
  - ALL<>, [406](#)
  - AreEqual< X, X >, [407](#)
  - AreEqual< X, Y >, [407](#)
  - compatible\_data\_type< Field >, [427](#)
  - compatible\_data\_type< Givaro::ZRing< double > >, [427](#)
  - compatible\_data\_type< Givaro::ZRing< float > >, [427](#)
  - ElementTraits< double >, [443](#)
  - ElementTraits< Element >, [442](#)
  - ElementTraits< FFPACK::rns\_double\_elt >, [443](#)
  - ElementTraits< float >, [443](#)
  - ElementTraits< Givaro::Integer >, [443](#)
  - ElementTraits< int16\_t >, [444](#)
  - ElementTraits< int32\_t >, [444](#)
  - ElementTraits< int64\_t >, [444](#)
  - ElementTraits< int8\_t >, [445](#)
  - ElementTraits< Reclnt::rint< K > >, [445](#)
  - ElementTraits< Reclnt::rmint< K, MG > >, [445](#)
  - ElementTraits< Reclnt::ruint< K > >, [446](#)
  - ElementTraits< uint16\_t >, [446](#)
  - ElementTraits< uint32\_t >, [446](#)
  - ElementTraits< uint64\_t >, [446](#)
  - ElementTraits< uint8\_t >, [447](#)
  - has\_minus\_eq\_impl< C >, [469](#)
  - has\_minus\_impl< C >, [470](#)
  - has\_mul\_eq\_impl< C >, [470](#)
  - has\_mul\_impl< C >, [470](#)
  - has\_operation< T >, [471](#)
  - has\_plus\_eq\_impl< C >, [471](#)
  - has\_plus\_impl< C >, [471](#)
  - is\_all\_same< T, Args... >, [478](#)
  - is\_all\_same<>, [478](#)
  - is\_simd< T >, [479](#)
  - ModeTraits< Field >, [510](#)
  - ModeTraits< Givaro::Modular< Element, Compute > >, [510](#)
  - ModeTraits< Givaro::Modular< Givaro::Integer, Compute > >, [511](#)
  - ModeTraits< Givaro::Modular< int16\_t, Compute > >, [511](#)
  - ModeTraits< Givaro::Modular< int32\_t, Compute > >, [511](#)
  - ModeTraits< Givaro::Modular< int64\_t, uint64\_t > >, [512](#)
  - ModeTraits< Givaro::Modular< int8\_t, Compute > >, [512](#)
  - ModeTraits< Givaro::Modular< Reclnt::ruint< K >, Compute > >, [512](#)
  - ModeTraits< Givaro::Modular< uint16\_t, Compute > >, [513](#)
  - ModeTraits< Givaro::Modular< uint32\_t, Compute > >, [513](#)
  - ModeTraits< Givaro::Modular< uint8\_t, Compute > >, [513](#)
  - ModeTraits< Givaro::ModularBalanced< Element > >, [514](#)
  - ModeTraits< Givaro::ModularBalanced< Givaro::Integer > >, [514](#)
  - ModeTraits< Givaro::ModularBalanced< int16\_t > >, [514](#)
  - ModeTraits< Givaro::ModularBalanced< int32\_t > >, [515](#)
  - ModeTraits< Givaro::ModularBalanced< int8\_t > >, [515](#)
  - ModeTraits< Givaro::Montgomery< T > >, [515](#)
  - ModeTraits< Givaro::ZRing< double > >, [516](#)
  - ModeTraits< Givaro::ZRing< float > >, [516](#)
  - ModeTraits< Givaro::ZRing< Givaro::Integer > >, [516](#)
  - need\_field\_characteristic< Field >, [517](#)
  - need\_field\_characteristic< Givaro::Modular< Field > >, [517](#)
  - need\_field\_characteristic< Givaro::ModularBalanced< Field > >, [517](#)
  - SimdChooser< T, false, b >, [732](#)
  - SimdChooser< T, true, false >, [733](#)
  - SimdChooser< T, true, true >, [733](#)
  - width< double >, [770](#)
  - width< float >, [770](#)
  - width< T >, [770](#)
  - vand
    - ScalFunctions< Element >, [552](#)
    - Simd128\_impl< true, true, false, 2 >, [571](#)
    - Simd128\_impl< true, true, false, 4 >, [581](#)
    - Simd128\_impl< true, true, false, 8 >, [592](#)
    - Simd128\_impl< true, true, true, 2 >, [601](#)
    - Simd128\_impl< true, true, true, 4 >, [610](#)
    - Simd128\_impl< true, true, true, 8 >, [619](#)
    - Simd128i\_base, [621](#)
    - Simd256\_impl< true, false, true, 8 >, [628](#)
    - Simd256\_impl< true, true, false, 4 >, [657](#)
    - Simd256\_impl< true, true, true, 4 >, [693](#)
    - Simd512\_impl< true, true, false, 8 >, [721](#)
    - Simd512\_impl< true, true, true, 8 >, [730](#)
    - Simd512i\_base, [732](#)
  - vandnot



- ScalFunctions< Element >, 553
- Simd128\_impl< true, true, false, 2 >, 572
- Simd128\_impl< true, true, false, 4 >, 582
- Simd128\_impl< true, true, false, 8 >, 592
- Simd128\_impl< true, true, true, 2 >, 601
- Simd128\_impl< true, true, true, 4 >, 610
- Simd128\_impl< true, true, true, 8 >, 619
- Simd128i\_base, 621
- Simd256\_impl< true, false, true, 8 >, 628
- Simd256\_impl< true, true, false, 4 >, 657
- Simd256\_impl< true, true, true, 4 >, 693
- Simd512\_impl< true, true, false, 8 >, 721
- Simd512\_impl< true, true, true, 8 >, 730
- Simd512i\_base, 732
- VEC\_ADD
  - FFLAS::vectorised, 284
- VEC\_SUB
  - FFLAS::vectorised, 284
- vect\_size
  - FieldSimd< \_Field >, 456
  - NoSimd< T >, 519
  - Simd128\_impl< true, true, false, 2 >, 572
  - Simd128\_impl< true, true, false, 4 >, 582
  - Simd128\_impl< true, true, false, 8 >, 592
  - Simd128\_impl< true, true, true, 2 >, 601
  - Simd128\_impl< true, true, true, 4 >, 610
  - Simd128\_impl< true, true, true, 8 >, 620
  - Simd256\_impl< true, false, true, 8 >, 629
  - Simd256\_impl< true, true, false, 2 >, 639
  - Simd256\_impl< true, true, false, 4 >, 657
  - Simd256\_impl< true, true, false, 8 >, 667
  - Simd256\_impl< true, true, true, 2 >, 676
  - Simd256\_impl< true, true, true, 4 >, 693
  - Simd256\_impl< true, true, true, 8 >, 702
  - Simd512\_impl< true, false, true, 8 >, 710
  - Simd512\_impl< true, true, false, 8 >, 721
  - Simd512\_impl< true, true, true, 8 >, 731
  - TestOneMethod< Simd >, 764
- vect\_t
  - FieldSimd< \_Field >, 451
  - NoSimd< T >, 518
  - Simd128\_impl< true, true, false, 2 >, 564
  - Simd128\_impl< true, true, false, 4 >, 574
  - Simd128\_impl< true, true, false, 8 >, 584
  - Simd128\_impl< true, true, true, 2 >, 594
  - Simd128\_impl< true, true, true, 4 >, 603
  - Simd128\_impl< true, true, true, 8 >, 612
  - simd128\_int64.inl, 875
  - Simd128i\_base, 620
  - Simd256\_impl< true, false, true, 8 >, 623
  - Simd256\_impl< true, true, false, 2 >, 632
  - Simd256\_impl< true, true, false, 4 >, 643
  - Simd256\_impl< true, true, false, 8 >, 660
  - Simd256\_impl< true, true, true, 2 >, 669
  - Simd256\_impl< true, true, true, 4 >, 680
  - Simd256\_impl< true, true, true, 8 >, 696
  - simd256\_int64.inl, 877
  - Simd256i\_base, 703
- Simd512\_impl< true, false, true, 8 >, 705
- Simd512\_impl< true, true, false, 8 >, 713
- Simd512\_impl< true, true, true, 8 >, 723
- simd512\_int64.inl, 880
- Simd512i\_base, 731
- TestOneMethod< Simd >, 763
- vectElt
  - ScalFunctions< Element >, 552
  - TestOneMethod< Simd >, 763
- verification\_PLUQ
  - benchmark-pluq.C, 798
- verifPLUQ
  - test-lu.C, 1087
- VERSION
  - config.h, 804
- vor
  - ScalFunctions< Element >, 553
  - Simd128\_impl< true, true, false, 2 >, 571
  - Simd128\_impl< true, true, false, 4 >, 581
  - Simd128\_impl< true, true, false, 8 >, 592
  - Simd128\_impl< true, true, true, 2 >, 601
  - Simd128\_impl< true, true, true, 4 >, 610
  - Simd128\_impl< true, true, true, 8 >, 619
  - Simd128i\_base, 621
  - Simd256\_impl< true, false, true, 8 >, 628
  - Simd256\_impl< true, true, false, 4 >, 657
  - Simd256\_impl< true, true, true, 4 >, 693
  - Simd512\_impl< true, true, false, 8 >, 721
  - Simd512\_impl< true, true, true, 8 >, 730
  - Simd512i\_base, 731
- vxor
  - ScalFunctions< Element >, 553
  - Simd128\_impl< true, true, false, 2 >, 572
  - Simd128\_impl< true, true, false, 4 >, 582
  - Simd128\_impl< true, true, false, 8 >, 592
  - Simd128\_impl< true, true, true, 2 >, 601
  - Simd128\_impl< true, true, true, 4 >, 610
  - Simd128\_impl< true, true, true, 8 >, 619
  - Simd128i\_base, 621
  - Simd256\_impl< true, false, true, 8 >, 628
  - Simd256\_impl< true, true, false, 4 >, 657
  - Simd256\_impl< true, true, true, 4 >, 693
  - Simd512\_impl< true, true, false, 8 >, 721
  - Simd512\_impl< true, true, true, 8 >, 730
  - Simd512i\_base, 731
- WAIT
  - parallel.h, 1031
- width< double >, 770
  - value, 770
- width< float >, 770
  - value, 770
- width< T >, 770
  - value, 770
- Winograd, 770
  - FFLAS::BLAS3, 197
- winograd.C, 774
  - balanced, 775
  - DOUBLE\_TO\_FLOAT\_CROSSOVER, 775

- GFOPS, [775](#)
  - main, [775](#)
  - TTimer, [775](#)
- Winograd\_L\_S
  - FFLAS::BLAS3, [200](#)
- Winograd\_LR\_S
  - FFLAS::BLAS3, [200](#)
- Winograd\_R\_S
  - FFLAS::BLAS3, [201](#)
- WinogradAcc\_2\_24
  - FFLAS::BLAS3, [198](#)
- WinogradAcc\_2\_27
  - FFLAS::BLAS3, [198](#)
- WinogradAcc\_3\_21
  - FFLAS::BLAS3, [198](#)
- WinogradAcc\_3\_23
  - FFLAS::BLAS3, [197](#)
- WinogradAcc\_L\_S
  - FFLAS::BLAS3, [200](#)
- WinogradAcc\_LR
  - FFLAS::BLAS3, [199](#)
- WinogradAcc\_R\_S
  - FFLAS::BLAS3, [199](#)
- WinogradCalc
  - FFLAS::Protected, [222](#)
- WinogradPar, [770](#)
- WinogradSteps
  - FFLAS::Protected, [221](#)
- WinogradThreshold
  - FFLAS::Protected, [220](#)
- WinoPar
  - FFLAS::BLAS3, [197](#)
- WINOTHRESHOLD
  - fflas.h, [826](#)
- WRITE
  - parallel.h, [1032](#)
- write
  - RNSInteger< RNS >, [542](#)
  - RNSIntegerMod< RNS >, [548](#)
- write\_field
  - Matio.h, [1047](#)
- write\_matrix
  - benchmark-fgemv-mp.C, [787](#)
  - RNSIntegerMod< RNS >, [548](#)
- write\_matrix\_long
  - RNSIntegerMod< RNS >, [548](#)
- writeCommandString
  - FFLAS, [188](#)
- writeDebugData
  - TestOneMethod< Simd >, [764](#)
- writeDnsFormat
  - FFLAS, [154](#)
- WriteMatrix
  - FFLAS, [188](#), [191](#)
- WritePermutation
  - FFLAS, [191](#)
- writeResultLine
  - TestOneMethod< Simd >, [764](#)

- zero
  - FFLAS, [76](#)
  - FieldSimd< \_Field >, [454](#)
  - RNSInteger< RNS >, [543](#)
  - RNSIntegerMod< RNS >, [550](#)
  - ScalFunctions< Element >, [552](#)
  - Simd128\_impl< true, true, false, 2 >, [571](#)
  - Simd128\_impl< true, true, false, 4 >, [581](#)
  - Simd128\_impl< true, true, false, 8 >, [591](#)
  - Simd128\_impl< true, true, true, 2 >, [601](#)
  - Simd128\_impl< true, true, true, 4 >, [609](#)
  - Simd128\_impl< true, true, true, 8 >, [619](#)
  - Simd128i\_base, [620](#)
  - Simd256\_impl< true, false, true, 8 >, [624](#)
  - Simd256\_impl< true, true, false, 2 >, [639](#)
  - Simd256\_impl< true, true, false, 4 >, [657](#)
  - Simd256\_impl< true, true, false, 8 >, [667](#)
  - Simd256\_impl< true, true, true, 2 >, [676](#)
  - Simd256\_impl< true, true, true, 4 >, [693](#)
  - Simd256\_impl< true, true, true, 8 >, [702](#)
  - Simd256i\_base, [703](#)
  - Simd512\_impl< true, false, true, 8 >, [705](#)
  - Simd512\_impl< true, true, false, 8 >, [721](#)
  - Simd512\_impl< true, true, true, 8 >, [730](#)
  - Simd512i\_base, [731](#)
- ZOSparseMatrix
  - FFLAS, [72](#)