Feb 2018

The Monte Carlo generator CASCADE: LHE interface and initial state parton showers from TMDs Version 3.0.01-beta01

H. Jung[1],
[1]DESY, Hamburg, FRG

Abstract

The interface to read in LHE files into the CASCADE package is described. The LHE files can be either files with off-shell initial state partons, as generated by the KATIE package or also files which are generated by collinear fixed order calculations like POWHEG or MADGRAPH, where a transverse momentum of the initial state partons is added according to the appropriate transverse momentum dependent (TMD) parton distribution.

PROGRAM SUMMARY

Title of Program: CASCADE 3.0.01-beta01

Computer for which the program is designed and others on which it is operable: any with standard Fortran 77 (g77 or gfortran), tested on SGI, HP-UX, SUN, PC, MAC

Programming Language used: FORTRAN 77

High-speed storage required: No

Separate documentation available: No

Keywords: QCD, TMD parton distributions.

Method of solution: Since measurements involve complex cuts and multi-particle final states, the ideal tool for any theoretical description of the data is a Monte Carlo event generator which generates initial state parton showers according to Transverse Momentum Dependent (TMD) parton densities, in a backward evolution. The evolution follows the DGLAP evolution equation exactly as used for the determination of the TMD.

Restrictions on the complexity of the problem: Any LHE file (with on-shell or off-shell) initial state partons can be processed.

Other Program used: PYTHIA (*version* > 6.4) for hadronization, TMDLIB as a library for TMD parton densities BASES/SPRING 5.1 for integration (supplied with the program package).

Download of the program: `http://www.desy.de/~jung/cascade`

Unusual features of the program: None

1 Introduction

In recent years the simulation of processes at the LHC has been often separated into two parts, the precision calculation of the hard process at higher order in the strong coupling α_s at next-to-leading (or even higher) order as implemented in packages like POWHEG [1,2] or MC@NLO [3–6] or HERWIG and the simulation of the subsequent parton radiation in form of parton showers and hadronization and multiparton interaction, which is then performed by PYTHIA or HERWIG. The interface between both parts is the so-called Les Houches Event (LHE) file [7], which contains all the information of the hard process including the color structure.

With the developments in determination of transverse momentum dependent (TMD) parton densities [8,9], it is natural to develop a scheme, where the initial parton shower follows exactly the TMD parton density and where either collinear (on-shell) of k_t-dependent (off-shell) hard process calculations can be combined. The Monte Carlo event generator CASCADE [10–12] is used for this, since it provides already a frame to perform initial state parton showers following the un-integrated gluon density. In this report we describe, how this frame can be extended to include all flavors in the parton shower and how the hard process can be used via LHE files.

2 The hard process

The hard process is generated externally either with POWHEG [1,2] with on-shell kinematics or with KATIE [13] with off-shell kinematics for the initial state partons. The events from the hard process are read into the CASCADE package via LHE files.

For processes generated with KATIE no further corrections need to be performed and the event can be directly passed to the showering procedure, described in the next section.

For processes with collinear kinematics of the initial state partons, a transverse momentum is added, according to the TMD parton density, however, care has to be taken, that energy and momentum is still conserved. The procedure is the following: for each initial parton, a transverse momentum is assigned according to the TMD density, and this system is rotated and booted to its center-of-mass frame. Since the initial state partons have transverse momentum, they acquire a virtuality. The energy and longitudinal component of the initial momenta are recalculated taken this virtuality into account, by [14]:

$$E_{1,2} = \frac{1}{\sqrt{2\hat{s}}}\left(\hat{s} \pm (Q_2^2 - Q_1^2)\right) \tag{1}$$

$$p_{z\;1,2} = \pm\frac{1}{2\sqrt{\hat{s}}}\sqrt{(\hat{s} + Q_1^2 + Q_2^2)^2 - 4Q_1^2 Q_2^2} \tag{2}$$

where Q_1^2 and Q_2^2 are the virtualities of parton $1,2$ after the transverse momentum is assigned. The final partons of the hard system are rotated and boosted to it's center-of-mass frame. Then the whole system of initial and final state partons is boosted and rotated back to its original system. This procedure is similar to the procedure applied in standard parton showers like PYTHIA, when a transverse momentum is created from the shower. The difference here is, that the transverse momentum is taken from the TMD directly, and the initial state shower will not change this anymore.

3 Initial State Parton Shower based on TMDs

The parton shower, which is described here, follows consistently the parton evolution of the TMDs. By this we mean that the splitting functions P_{ab}, the order in $\alpha_{\rm s}$, the scale in the calculation of $\alpha_{\rm s}$ as well as the kinematic restrictions applied are identical in both the parton shower and the evolution of the parton densities.

A backward evolution method, as now common in Monte Carlo event generators, is applied for the initial state parton shower, evolving from the large scale of the matrix-element process backwards down to the scale of the incoming hadron. However, in contrast to the conventional parton shower, which generates a transverse momentum of the initial state partons during the backward evolution, the transverse momentum of the initial partons of the hard scattering process is fixed by the TMD and the parton shower does not change the kinematics. The transverse momenta during the cascade follow the behavior of the TMD. The hard scattering process is obtained directly using off-shell matrix element calculations as described in section 2.

The backward evolution of the initial state parton shower follows very closely the description in [10–12, 14]. The evolution scale μ is selected from the hard scattering process, with $\mu^2 = \hat{p}_T^2$ or $\mu^2 = Q_t^2 + \hat{s}$ for an evolution in virtuality or angular ordering, with $\hat{p}_T$ being the transverse momentum of the hard process, Q_t being the vectorial sum of the initial state transverse momenta and s being the invariant mass of the subprocess.

Starting with the hard scale $\mu = \mu_i$, the parton shower algorithm searches for the next scale μ_{i-1} at which a resolvable branching occurs. This scale μ_{i-1} is selected from the Sudakov form factor Δ_S making use of the TMD densities $\mathcal{A}_a(x', k_t', \mu')$ which depend on the longitudinal momentum fraction $x' = xz$ of parton a, its transverse momentum k_t' probed at a scale μ' (see also [10]). The Sudakov form factor Δ_S for the backward evolution is given by (see fig. 1 left):

$$\Delta_S(x, \mu_i, \mu_{i-1}) = \exp\left[-\int_{\mu_{i-1}}^{\mu_i} \frac{d\mu'}{\mu'} \frac{\alpha_{\rm s}(\tilde{\mu}')}{2\pi} \sum_a \int dz P_{a\to bc}(z) \frac{x'\mathcal{A}_a(x', k_t', \mu')}{x\mathcal{A}_b(x, k_t, \mu')} \right] \qquad (3)$$

which describes the probability that parton b remains at x with transverse momentum k_t when evolving from μ_i to $\mu_{i-1} < \mu$. Please note, that the argument in $\alpha_{\rm s}$ is $\tilde{\mu}'$ and depends on the ordering condition as discussed later. [1]

In the parton shower language, the selection of the next branching comes from solving the Sudakov form factor eq.(3) for μ_{i-1}. However, to solve the integrals in eq.(3) numerically for

[1] In equation eq.(3) ordering in μ is assumed, if angular ordering, as in CCFM [15–18], is applied then the ratio of parton densities would change to $\frac{x'\mathcal{A}_a(x', k_t', \mu'/z)}{x\mathcal{A}_b(x, k_t, \mu')}$ as discussed in [10].

every branching would be too time consuming, instead the veto-algorithm [14,19] is applied. The selection of μ_{i-1} and the branching splitting z_{i-1} follows the standard methods [14].

The splitting function P_{ab} as well as the argument $\tilde{\mu}$ in the calculation of α_s is chosen exactly as used in the evolution of the parton density. In a parton shower one treats "resolvable" branchings, defined via a cut in $z < z_M$ in the splitting function (see eq.(**??**)) to avoid the singular behavior of the terms $\frac{1}{1-z}$, and branchings with $z > z_M$ are regarded as "non-resolvable" and are treated similarly as virtual corrections: they are included in the Sudakov form factor Δ_S.

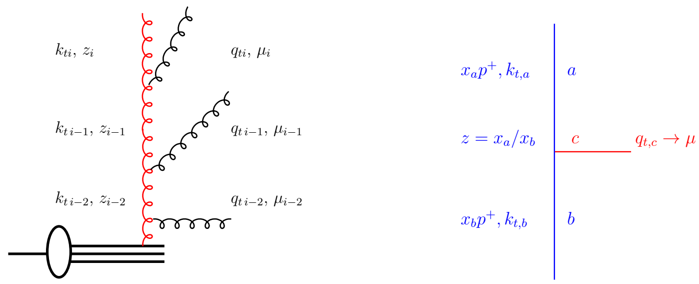

Figure 1: Left: Schematic view of a parton branching process. Right: Branching process $b \to a + c$.

The longitudinal momentum fraction $x_{i-1} = \frac{x_i}{z_{i-1}}$ is calculated by generating z_{i-1} according to the splitting function. With z_{i-1} and μ_{i-1} all variables needed for a collinear parton shower are obtained.

The calculation of the transverse momentum k_t is sketched in fig. 1 right. The transverse momentum $q_{t\,i}$ can be obtained by giving a physical interpretation to the evolution scale μ_i (see fig. 1 right), and $q_{t\,i}$ can be calculated in case of angular ordering (μ is associated with the angle of the emission) in terms of the angle Θ of the emitted parton wrt the beam directions $q_{t,c} = (1-z)E_b \sin\Theta$:

$$\mathbf{q}_{t,i}^2 = (1-z)^2 \mu_i^2 \quad . \tag{4}$$

Once the transverse momentum of the emitted parton q_t is known, the transverse momentum of the propagating parton can be calculated from

$$\mathbf{k}_{t\,i-1} = \mathbf{k}_{t\,i} + \mathbf{q}_{t\,i-1} \tag{5}$$

with a uniformly distributed azimuthal angle ϕ is assumed for the vector components of $\mathbf{k}$ and $\mathbf{q}$.

The whole procedure is iterated until one reaches a scale $\mu_{i-1} < q_0$ with q_0 being a cut-off parameter, which can be chosen to be the starting evolution scale of the TMD. However, it turns out that during the backward evolution the transverse momentum k_t can reach large values, even for small scales μ_{i-1}, because of the random ϕ distribution. On average the transverse momentum decreases, and it is of advantage to continue the parton shower evolution to a scale $q_0 \sim \Lambda_{qcd} \sim 0.3$ GeV, to allow enough emissions to share the transverse momenta generated.

3.1 The TMD parton density

In the previous versions of CASCADE the TMD densities where part of the program. With the development of TMDLIB [20] there is easy access to all available TMDs, they can be selected, as before, via `IGLU` with a value > 100000. For example the TMDs from the parton branching method [8,9] are selected via `IGLU=101600` or the ones from the KMR approach, as used in are selected via `IGLU=410000` as in Ref. [21]

3.2 α_s

The strong coupling α_s is calculated via the PYTHIA [?] subroutine `PYALPS`. Maximal and minimal number of flavours used in α_s are set by `MSTU(113)` and `MSTU(114)`, Λ_{QCD} = `PARU(112)` with respect to the number of flavours given in `MSTU(112)` .

3.3 Final state parton showers

The final state parton shower uses the parton shower routine `PYSHOW` of PYTHIA with the default scale $\mu^2 = 2 \cdot (m^2_{1\,\perp} + m^2_{2\,\perp})$ (`IFIN=1`), with $m_{1(2)\,\perp}$ being the transverse mass of the hard parton 1(2). Other choices are possible: $\mu^2 = \hat{s}$ (`IFIN=2`) and $\mu^2 = 2 \cdot (m^2_1 + m^2_2)$ (`IFIN=3`). Relevant for processing LHE files is the scale `SCALUP` provided from the hard scattering process. This scale can be used for final state parton shower with `IFIN=4` In addition a scale factor can be applied: `SCAF`$\times\mu^2$ (default: `SCAF=1`).

4 Description of the program components

In CASCADE all variables are declared as `Double Precision`. The

The program has to be compiled and linked together with PYTHIA 6, to ensure that the double precision code of JETSET is loaded.

When `HEPMC` is included, the output of CASCADE is a standard HEPMC [22] file, which can be further processed for example with Rivet [23].

4.0.1 Parameters

`NEVENT:`	number of events to be processed, for NEVENT = -1 all event in the LHE file are read.
`IPRO:`	=-1 read LHE file

4.0.2 Parameters for parton shower and fragmentation

`NFRAG:`	(D: = 1) switch for fragmentation = 0: off = 1: on
`IFPS:`	(D: = 3) switch for parton shower . = 0: off = 1: initial state = 2: final state = 3: initial and final state
`ITIM:`	(D: =1) =0: no shower of time like partons =1: time like partons may shower
`ICCFM:`	(D: =1) =0: DGLAP type evolution (one loop, old version) =1: CCFM evolution (all loops) =2: full flavor TMD parton shower with DGLAP splitting function. No upper cut on $k_t$in shower applied =3: full flavor TMD parton shower with DGLAP splitting function with upper cut on $k_t$of shower given by $k_t$of off-shell initial partons
`IFIN`	(D:=1) scale switch for final state parton shower = 1: $\mu^2 = 2(m_{1\ t}^2 + m_{2\ t}^2)$ = 2: $\mu^2 = \hat{s}$ = 3: $\mu^2 = 2(m_1^2 + m_2^2)$ = 4: 2*`SCALUP`
`SCAF`	(D:=1.) scale factor for final state parton shower

4.0.3 Parameters for parton densities

`IGLU:`	(D: = 1201) select TMD parton density > 100000: call TMDLIB for TMD densities `IGLU=410000` BHKS TMD [21] `IGLU=101600` parton branching TMD [8]

4.0.4 Parameters for LHE files

`CLHE`	name of LHE file
`ITMW`	=0: LHE file has off shell initial partons = 1: generate $k_t$according to TMD given with IGLU and change initial on-shell partons = 2: same as =2, but reweight also to TMD given with IGLU but with fact.scale

5 Program Installation

CASCADE now follows the standard AUTOMAKE convention. To install the program, do the following

```
1) Get the source

tar xvfz cascade-XXXX.tar.gz
cd cascade-XXXX

2) Generate the Makefiles (do not use shared libraries)
./configure --disable-shared --prefix=install-path --with-pythia6="pythia_path"  --with-lhapdf="lhapdflib_path"
--with-tmdlib="TMDlib-path" --with-gsl="gsl_lib" --with-hepmc="hepmc_path"

with (as example):
pythia_path=/afs/desy.de/group/alliance/mcg/public/MCGenerators/pythia6/422/i586_rhel40
lhapdflib_path=/afs/desy.de/group/alliance/mcg/public/MCGenerators/lhapdf/5.8.1/i586_rhel40

3) Compile the binary
make

4) Install the executable and PDF files
make install

4) The executable is in bin
run it with:
export CASED=1242425
export HEPMCOUT=outfile.hepmc

cascade < steer_pp-LHEin
```

References

[1] S. Alioli, K. Hamilton, P. Nason, C. Oleari, and E. Re, JHEP **04**, 081 (2011). 1012.3380.

[2] S. Frixione, P. Nason, and C. Oleari, JHEP **0711**, 070 (2007). 0709.2092.

[3] S. Frixione and B. R. Webber (2006). hep-ph/0612272.

[4] S. Frixione, P. Nason, and B. R. Webber, JHEP **08**, 007 (2003). hep-ph/0305252.

[5] S. Frixione and B. R. Webber (2002). hep-ph/0207182.

[6] S. Frixione and B. R. Webber, JHEP **0206**, 029 (2002). hep-ph/0204244.

[7] J. Alwall *et al.*, Comput. Phys. Commun. **176**, 300 (2007). hep-ph/0609017.

[8] F. Hautmann, H. Jung, A. Lelek, V. Radescu, and R. Zlebcik (2017). 1708.03279.

[9] F. Hautmann, H. Jung, A. Lelek, V. Radescu, and R. Zlebcik, Phys. Lett. **B772**, 446 (2017). 1704.01757.

[10] H. Jung, S. Baranov, M. Deak, A. Grebenyuk, F. Hautmann, *et al.*, Eur.Phys.J. **C70**, 1237 (2010). 1008.0152.

[11] H. Jung, Comput. Phys. Commun. **143**, 100 (2002). hep-ph/0109102.

[12] H. Jung and G. P. Salam, Eur. Phys. J. **C19**, 351 (2001). hep-ph/0012143.

[13] A. van Hameren (2016). `1611.00680`.

[14] M. Bengtsson, T. Sjostrand, and M. van Zijl, Z. Phys. **C32**, 67 (1986).

[15] M. Ciafaloni, Nucl. Phys. **B296**, 49 (1988).

[16] S. Catani, F. Fiorani, and G. Marchesini, Phys. Lett. **B234**, 339 (1990).

[17] S. Catani, F. Fiorani, and G. Marchesini, Nucl. Phys. **B336**, 18 (1990).

[18] G. Marchesini, Nucl. Phys. **B445**, 49 (1995). `hep-ph/9412327`.

[19] S. Platzer and M. Sjodahl, Eur.Phys.J.Plus **127**, 26 (2012). `1108.6180`.

[20] F. Hautmann, H. Jung, M. Krämer, P. Mulders, E. Nocera, *et al.*, Eur. Phys. J. **C 74**, 3220 (2014). `1408.3015`.

[21] M. Bury, A. van Hameren, H. Jung, K. Kutak, S. Sapeta, and M. Serino (2017). `1712.05932`.

[22] M. Dobbs and J. B. Hansen, Comput. Phys. Commun. **134**, 41 (2001).

[23] A. Buckley, J. Butterworth, L. Lonnblad, D. Grellscheid, H. Hoeth, J. Monk, H. Schulz, and F. Siegert, Comput. Phys. Commun. **184**, 2803 (2013). `1003.0694`.