A guide to the ANU p-Quotient Program Version 1.9

Eamonn O'Brien

January 2012

Contents

Abstract

The ANU p-Quotient Program (pq) provides access to implementations of an algorithm to construct a power-commutator presentation (pcp) for a p-group and of an algorithm to generate descriptions of p-groups. It also allows access to an implementation of an algorithm which can be used to construct a "canonical" pcp for a p-group and via this construction it allows a user to determine whether two p-groups are isomorphic. The latter can be used to generate a description of its automorphism group.

1 Program content

The ANU p-Quotient Program (pq) is named for the p-quotient algorithm that it first implemented. Now, via menus it provides access to implementations of all the following algorithms:

1. A *p-quotient algorithm* to compute a power-commutator presentation for a p-group. The algorithm implemented here is based on that described in Newman and O'Brien (1996), Havas and Newman (1980), and papers referred to there.

   Another description of the algorithm appears in Vaughan-Lee (1990b). A FORTRAN implementation of this algorithm was programmed by Alford & Havas. The basic data structures of that implementation are retained.

   The current implementation incorporates the following features:

   a. collection from the left (see Vaughan-Lee, 1990b); Vaughan-Lee's implementation of this collection algorithm is used in the program;
   b. an improved consistency algorithm (see Vaughan-Lee, 1982);
   c. new exponent law enforcement and power routines;
   d. closing of relations under the action of automorphisms;
   e. some formula evaluation.

2. A *p-group generation algorithm* to generate descriptions of p-groups. The algorithm implemented here is based on the algorithms described in Newman (1977) and O'Brien (1990). A FORTRAN implementation of this algorithm was earlier developed by Newman & O'Brien.

3. A *standard presentation algorithm* used to compute a canonical power-commutator presentation of a p-group. The algorithm implemented here is described in O'Brien (1994).

4. An algorithm which can be used to compute the *automorphism group* of a p-group. The algorithm implemented here is described in O'Brien (1995).

The pq program is written in traditional C and contains about 22000 lines of code. It was developed in a SUN OS environment and has been ported successfully to each of Solaris, AIX and Ultrix environments. The interface and input/output facilities of the program are rudimentary. Interfaces have been developed which allow parts of this program to be called from within the computational group theory system GAP. This

program is supplied as a package within GAP. The link from GAP to pq is described in the manual for GAP 3.4 or in the manual found in the `doc` directory in the case of the GAP 4 package ANUPQ; all of the necessary code can be found in the `gap` directory of these distributions. The program is also distributed as part of Quotpic.

Version 1.9 of the pq program (i.e. the version you are currently reading the documentation for) is configured to call GAP, at least version 4.5, to compute stabilisers when needed.

The FORTRAN version of this program was known as the Nilpotent Quotient Program.

2 Installing pq

To install pq just follow the instructions for installing the ANUPQ package, i.e. in the directory above, do:

```
./configure
make TARGET GAP=GAPPATH
```

If you are running Linux you may omit *TARGET*. The path *GAPPATH* should be the path of a *shell script* that runs GAP 4 with packages AutPGrp (at least version 1.5) and ANUPQ (at least version 3.1) installed. If the ANUPQ and/or AutPGrp packages are not installed in the `pkg` directory of the GAP 4 distribution, then the `-l` option (last line of the *shell script* previously mentioned) *must* be set to be a semicolon-separated list of paths of the main GAP directory and the path of the `pkg` directory containing those packages of ANUPQ and AutPGrp that are missing from the GAP 4 distribution `pkg` directory.

The pq program only needs GAP for computing stabilisers when you answer "No" (`0`) to the question "`PAG-generating sequence for automorphism group?`". If you neglect to add: "`GAP=`*GAPPATH*" to the `make` command, never mind; the pq program first checks to see if the environment variable `ANUPQ_GAP_EXEC` is set. For `csh` doing:

```
setenv ANUPQ_GAP_EXEC GAPPATH
```

is essentially equivalent to including: `GAP=`*GAPPATH* in the `make` command. If you use a different shell just use the appropriate syntax for setting environment variables for your shell.

A good test for checking that you have installed pq correctly is to try the example in Appendix A.2.

3 Organisation

Access to the implementations of each algorithm mentioned in Section 1 is provided via menus. The *p-quotient algorithm* machinery is provided by the "Basic Menu for p-Quotient Program" and the "Advanced p-Quotient Menu"; access to the *p-group generation algorithm* implementation is provided via the main "Menu for p-Group Generation" and the "Advanced Menu for p-Group Generation"; and finally the *standard presentation algorithm* and *automorphism group algorithm* implementations may be accessed via the "Standard Presentation Menu".

By default, the pq program opens with the Basic Menu [for p-Quotient Program] (see 6.1), which provides "basic" options for the p-Quotient algorithm and via options `8` and `9` gives (immediate) access to two further menus, namely the Advanced p-Quotient Menu (see 6.2) and the (main) Menu for p-Group Generation (see 7.4), respectively. The main Menu for p-Group Generation provides access to a further menu: the Advanced Menu for p-Group Generation (see 7.6).

If the pq program is invoked with the `-i` switch (see Section 4), the program opens with the Standard Presentation Menu (see Section 8) which gives one immediate access to the standard presentation and automorphism group machinery and via option `7` access to the Basic Menu and hence the other menus.

To cleanly exit the pq program one must retrace one's path through the menus (by typing the number of the option that exits each menu or `0`). Exiting the menu that the pq program opened with (either the Basic Menu or the Standard Presentation Menu), exits the pq program.

The Basic Menu, the main Menu for p-Group Generation and the Standard Presentation Menu are designed particularly for the new or occasional user; levels of control by the user are low, but there is little attendant risk of obtaining inaccurate information. The Advanced menus are designed for the knowledgeable user with specialised needs; they allow the user to make almost all decisions, provide little protection and if pre-requisite information for an option to succeed (e.g. a pc presentation for the group) is not in place, the pq will invariably exit horribly.

Each menu is discussed in more detail later. See Section 5 for the conventions that apply to these menus.

4 Runtime parameters

The program may be invoked with the following runtime parameters:

`-b` A "basic" format can be used to input a group presentation. See 5.2.2.

`-G` This option is used by GAP 4. It is essentially equivalent to setting the switches `-g -i -k` simultaneously, except that it uses GAP's iostream to direct requests to GAP to compute stabilisers when necessary.

`-g` If groups are generated using p-group generation, then their presentations are written to a file in a GAP compatible format. The name of the file may be selected using the `-w` option; the default is `GAP_library`.

`-i` This provides access to the Standard Presentation Menu, which can be used to construct the standard presentation of a given p-group.

`-k` The presentation may be defined and supplied using certain key words. Examples of this format can be found in those files in the `examples` directory whose names commence with `keywords_`. This option cannot be used with `-b`.

`-s` *integer*
All computations of power-commutator presentations occur in an integer array, y – the space of this array, by default 1000000, is set to *integer*. See the discussion on strategies to minimise time and space later in this document.

`-v` Gives the version of the ANU p-Quotient program and exits.

`-w` *file*
Group descriptions are written in GAP format to *file*. `-g` must be used in conjunction with this parameter.

If the program is compiled using the `RUN_TIME` option, then there are two additional runtime options:

`-c` The maximum exponent-p class to be considered.

`-d` A bound on the number of defining generators.

5 Conventions

5.1 Menu Conventions

The following conventions apply for all menus.

- Typing the integer identifying an option selects that option.
- At a number of points in running the program, you will be asked questions or prompted for input. A non-zero integer response signifies a positive response to the question; a response of `0` (zero) is a negative response. In this guide, a "Yes" means a non-zero integer response; a "No" is a zero response.
- For each menu, the option `-1` lists the menu, and `0` exits the menu. To cleanly exit the pq program one must retrace one's path through the menus (by typing the number of the option that exits each menu or `0`). Exiting the menu that the pq program opened with (either the Basic Menu or the Standard Presentation Menu), exits the pq program.
- If the program cannot open a file for any reason, it simply reports this and if not running interactively exits.
- Input from the first occurrence of a "`#`" symbol to the end of that line is interpreted as a comment and is ignored.

Language used in the menus for the construction of the pcp follows that used in Havas & Newman (1980) and Newman & O'Brien (1996); that in the p-group generation menus follows O'Brien (1990); that in the standard presentation menu follows O'Brien (1994).

5.2 Word input

The performance of the program is significantly enhanced if it can store the defining relations in their unexpanded form. It is currently only possible to store a supplied relation in its unexpanded form if the relation is either a power OR a commutator, and the components of the power or commutator are only defining generators or their inverses. Hence, it is frequently appropriate for the user to introduce redundant generators into the supplied presentation.

There are two formats available for supplying generators and relations: the "default format" (see Section 5.2.1) and the "basic format" (see Section 5.2.2). Examples of the usage of both formats can be found in the `examples` directory.

5.2.1 The default format for word input

Under the default format, a user is prompted for a list of generators, which must be supplied as a set. The user is then prompted for a defining set of relations, again supplied as a set.

Any combination of relations and relators may be supplied. Note, however, you may NOT use relations of the form $u = v = w$. Relations are separated by commas, $\wedge$ is used for powers and conjugation, `[` and `]` are used to indicate the beginning and end of a commutator, and `1` is the identity of the group. The following are examples of valid input for relations:

```
{x ∧ 5 = [x, y, y], z ∧ x = 1, z * y * x * z ∧ -1 = 1}
{a3 * [a2, a1], a4 ∧ a1 * a3 ∧ a2 * [a2, a1 ∧ -1, a4, a1 ∧ 7] = 1}
```

5.2.2 The basic format for word input

The basic format is selected at start-up via the `-b` switch (see Section 4). In the pcp, the defining generators and the pcp generators are labeled as positive integers; in each case they commence at 1. Inverses of generators are labelled by the corresponding negative number.

The format for word input is the following:

$$Exp\ Gen_1\ Gen_2 \ldots ;$$

where "*Exp*" is the exponent; if Gen_i is a positive integer, it represents the corresponding generator of the group; if it is a negative integer, it represents the inverse of that generator. Word input is terminated by a "`;`". Entries in the word can be separated by any positive number of spaces.

Defining relations may only be supplied as relations – not as relators. Each side of the relation is supplied as a word using the above format. Where the input is a power of a left-normed commutator, the following simpler format may be used

$$Exp\ \texttt{[}Gen_1\ Gen_2 \ldots \texttt{]};$$

where `[` and `]` are used to indicate the beginning and end of the commutator. As for the default format, entries in the commutator can be separated by an optional number of spaces. The identity word is indicated by supplying a word with exponent `0` – "`0;`" is sufficient.

Examples of acceptable input are the following:

- The input "`5 2 1 -3 4;`" represents the word $(2 \times 1 \times 3^{-1} \times 4)^5$.
- The input "`3 [2 1 1 3];`" represents the commutator power $[2,\ 1,\ 1,\ 3]^3$.

Under the basic format, the program only accepts input of either type in a word; you may not combine them. This may affect how you supply the defining relations for the presentation.

5.2.3 Advanced menu word input

Words are supplied as input to options to the Advanced menus on a number of occasions. Usually, these are words in the pcp generators of the group.

Under the default format, the n pcp generators of the group are, for convenience, automatically labelled as $\texttt{x1}, \texttt{x2}, \ldots, \texttt{x}n$. All words in the pcp generators are then supplied as words in $\texttt{x1}, \ldots, \texttt{x}n$, using the format prescribed above for the defining relators. Word input is terminated by a "`;`".

A few options allow input involving the defining generators. The m defining generators of the group are also automatically labelled as $\texttt{x1}, \texttt{x2}, \ldots, \texttt{x}m$. All words in the defining generators are then supplied as words in $\texttt{x1}, \ldots, \texttt{x}m$, using the format prescribed above for the defining relators. As before, word input is terminated by a "`;`".

If you use the basic input format, then all words are supplied as specified in the basic format discussion.

5.3 Input and output facilities

Currently, facilities exist to save the computed presentation to file and to later restore and restart the computation. The files are saved and restored using the "`fread`" and "`fwrite`" facilities, respectively. For both save and restore, the user supplies a name for the file, which can be any valid (UNIX or VMS) name. In the case of writing to file, the program does not query the user before overwriting an existing file – it is the user's responsibility to prevent such situations from occurring.

6 The p-Quotient implementation

6.1 Basic Menu for p-Quotient Program

The default opening menu obtained on running the program is listed below.

```
Basic Menu for p-Quotient Program
---------------------------------
1. Compute pc presentation
2. Save presentation to file
3. Restore presentation from file
4. Display presentation of group
5. Set print level
6. Calculate next class
7. Compute p-covering group
8. Advanced p-quotient menu
9. (Main) menu for p-group generation
10. Exit from p-quotient program
```

We now discuss each of these options.

`1. Compute pc presentation`

When you select this option, you will be given the following sequence of prompts for input. [If you start the pq program with the `-k` switch (see Section 4), you will not be prompted; instead you must supply input in the form:

keyword value

where *value* is the input you would otherwise have been prompted for. The last input for this option must also be terminated by a semicolon (`;`). Some data have default values if omitted, and there is no restriction on the order in which the data are supplied. The keyword *keyword* needed and default value if there is one, for the corresponding `-k` input is given in square brackets after describing each prompt (but not after describing the prompts obtained when the pq program is called with `-b` — the `-k` and `-b` switches cannot be used together). Actually, only the first 3 letters of each keyword are significant; so, in fact, `prime`, for example, may be abbreviated to `pri`.]

`Input group identifier:`

you may supply any sequence of characters, excluding spaces, as a valid identifier for the group. [keyword: `name`, default: `G`.]

`Input prime:`
supply the prime p used in computing the p-quotient. [keyword: `prime`.]

`Input maximum class:`
supply the maximum exponent-p class of the quotient to be constructed. [keyword: `classbound`, default: `10`.]

`Input print level:`
see option `5` below. [keyword: `outputlevel`, default: `1`.]

If the default format is used (i.e. pq was not called with the `-b` switch) then you will be prompted as follows.

`Input generating set (in { }):`
supply generating set. [keyword: `generators`.]

`Input defining set of relations (in { }):`
supply defining set of relations. [keyword: `relators`, default: `{}`.]

Otherwise, if the basic format is used (i.e. pq was called with the `-b` switch; see Section 4), you will not be prompted; instead you must supply input in the) then you will be given the following prompts.

`Input number of generators:`
supply number of defining generators.

`Input number of relations:`
supply number of defining relations.

Then (i.e. with or without the `-b` switch) you will be prompted

`Input exponent law (0 if none):`
if the group is to satisfy a particular exponent law, supply a positive value for that exponent. [keyword: `exponentlaw`, default: `0`.]

In the basic format case, you will finally also be prompted to input each relation for the group (if any).

2. `Save presentation to file`
prompts for file name, saves group presentation to file.

3. `Restore presentation from file`
prompts for file name, restores group presentation from that file if it exists.

4. `Display presentation of group`
displays group presentation; detail depends on print level; if level is one, then display order of group, otherwise display full pcp.

5. `Set print level`
ranges from 0, providing no output, to 3; 1, the default, provides minimal output.

6. `Calculate next class`
calculates pcp for quotient having one class greater than the class of the existing group.

7. `Compute p-covering group`

`8. Advanced p-quotient menu`
: provides access to the "Advanced Menu" (intended for experts) for user manipulation of the presentation.

`9. (Main) menu for p-group generation`
: provides access to the main menu for p-group generation.

`10. Exit from p-quotient program`
: causes the pq program to exit, or if pq was called with the `-i` switch (see Section 4) exits to the Standard Presentation Menu. Selecting option `0` performs the same function.

6.2 Advanced p-Quotient Menu

The advanced p-quotient menu, given below, is selected by choosing option `8` from the Basic Menu (see 6.1).

```
Advanced p-Quotient Menu
------------------------
1. Do individual collection
2. Solve the equation ax = b for x
3. Calculate commutator
4. Display group presentation
5. Set print level
6. Set up tables for next class
7. Insert tails for some or all classes
8. Check consistency for some or all classes
9. Collect defining relations
10. Carry out exponent checks
11. Eliminate redundant generators
12. Revert to presentation for previous class
13. Set maximal occurrences for pcp generators
14. Set metabelian flag
15. Carry out an individual consistency calculation
16. Carry out compaction
17. Carry out echelonisation
18. Supply and/or extend automorphisms
19. Close relations under automorphism actions
20. Print structure of a range of pcp generators
21. Display automorphism actions on generators
23. Collect word in defining generators
24. Compute commutator of defining generators
25. Write presentation to file in GAP format
26. Write compact description of group to file
27. Evaluate certain formulae
28. Evaluate action specified on defining generators
29. Evaluate Engel (p - 1)-identity
30. Process contents of relation file
31. Exit to basic menu
```

7 The p-Group Generation implementation

7.1 Required input

The required input is the p-covering group of the starting group, together with a description of the automorphism group of the p-covering group. Before you commence to construct descendants, you should construct or restore its p-covering group, using the appropriate options of the Basic Menu (see 6.1). It is the user's responsibility to do this – no check is performed.

7.2 The automorphism group description

You must also supply a description of its automorphism group, which is done by selecting option `1` of either of the menus for p-group generation (see 7.4 or 7.6). This description is the action of each automorphism on each of the pcp generators of the Frattini quotient of the group. The action is described by a vector of exponents – the length of the vector is the number of pcp generators of the group, its entries are the powers of each of these generators which occur in the image.

Where the automorphism group is soluble, a PAG-generating system should be supplied which works up a composition series for the group via cyclic factors of prime order. In such cases, the calculations may be carried out completely within pq. If the soluble group machinery is selected by the user, but a PAG-generating system is not supplied, then the program will give wrong information.

If the automorphism group is insoluble or a PAG-generating sequence is not supplied, a call is made by the program to one GAP, which computes stabilisers of particular orbit representatives. If the automorphism group of any of the intermediate (reduced) p-covering groups is found to be soluble, a PAG-generating sequence is computed by GAP and handed back to pq. The soluble machinery is now automatically invoked for the rest of the computation.

7.3 Saving group descriptions

The constructed groups of a particular class, c, are saved to a file, whose name is obtained by concatenating: *starting-group-identifier* and c, where *starting-group-identifier* is the group identifier defined at option `1` of the Basic Menu (see Section 6.1) e.g. if you use the `-k` switch (see Section 4) when you started the pq program and settled for the default group identifier `G` then for class 2, the groups will be saved to a file with name: `G_class2`. As before, the program does not query the user before overwriting an existing file.

7.4 The (Main) Menu for p-Group Generation

The main Menu for p-Group Generation, given below, is selected by choosing option `9` from the Basic Menu (see 6.1).

```
Menu for p-Group Generation
---------------------------
1. Read automorphism information for starting group
2. Extend and display automorphisms
3. Specify input file and group number
4. List group presentation
5. Construct descendants
6. Advanced p-group generation menu
7. Exit to basic menu
```

We now describe the options given by this menu.

`1. Read automorphism information for starting group`
first prompts for the following:

`Input the number of automorphisms:`
You must provide the number of automorphisms generating the automorphism group of the starting group. Then you will be prompted for the action of each automorphism on the pcp generators of the Frattini quotient of the group, after which you will be prompted:

`Input number of soluble generators for automorphism group:`
If you enter a positive integer n you will then be prompted for the relative order of each of those n automorphisms.

`2. Extend and display automorphisms`
compute the extensions of these automorphisms to act on the pcp generators of the p-covering group and display the results.

`3. Specify input file and group number`
prompts for the input file name and the group number.

`4. List group presentation`
display at output level 3 the presentation for the group.

`5. Construct descendants`
we discuss this option in detail in Section 7.5.

`6. Advanced p-group generation menu`
provides access to the "Advanced Menu" for user-controlled construction and manipulation.

`7. Exit to basic menu`
returns the user to the Basic Menu (see 6.1). Selecting option `0` performs the same function.

7.5 Construct descendants option

Here we discuss option `5` of the (main) p-group generation menu (see 7.4). If you select this option you receive a number of questions or prompts for input. Those prompts/questions are as follows.

`Input class bound on descendants:`

you must supply a positive integer greater than the class of the starting group, and which is an upper bound on the class of the descendants constructed.

`Construct all descendants?`

If you enter a "Yes" response (by entering a non-zero integer) you are asked:

`Set an order bound for descendants?`

If you answer "Yes" you are further prompted (if you answer "No" (i.e. `0`) you are not so prompted):

`Input order bound on descendants:`

and the integer order bound you input will apply in addition to the class bound selected.

If you responded "No" to the "`Construct all descendants?`" query and the class increase is one you are prompted:

`Input step size:`

and a positive integer is expected.

If you responded "No" to the "`Construct all descendants?`" query and the class increase is greater than one you are prompted as follows:

`Constant step size?`

If you answer "Yes" you are prompted:

`Input step size:`

and a positive integer is expected.

If you answer "No" to "`Constant step size?`" you are prompted:

`Input` n `step sizes:`

for some integer n, and you must enter n positive integers separated by spaces.

`PAG-generating sequence for automorphism group?`

This determines the algorithm used in constructing the orbits and stabilisers of representative allowable subgroups (see 7.7). Whether you answer "Yes" or "No" you are then asked:

`Default algorithm?`

A "Yes" here constructs immediate descendants using the smallest possible characteristic initial segment subgroups in the p-multiplicator. This minimises the degree of the permutation group constructed.

If you answer "No", then you will be prompted/asked:

`Rank of the initial segment subgroup?`

If you want to proceed as in the default, respond "`0`"; otherwise any positive value is accepted. If the value is larger than the rank of the p-multiplicator, the program takes this upper bound as the selected value. The initial segment subgroup determined by this response is characteristically closed by the program.

If your answer to "`PAG-generating sequence for automorphism group?`" was a "Yes", then you will receive the following further question.

`Space efficient computation?`

By default, all of the permutations constructed are stored. If you answer "Yes", at any one time only one permutation is stored, consequently reducing the amount of space significantly. However, the time taken in computing the automorphism group of each descendant is also significantly increased.

Then you will prompted/asked the following:

`Completely process terminal descendants?`

By default, automorphism groups are computed for and descriptions of capable groups only are saved to file. If you wish to compute automorphism groups and save descriptions of all descendants to file, then answer "Yes". In this case, for both terminal and capable groups, the automorphism group is saved to file; if capable, the pcp of the p-covering group is saved; if terminal, the pcp for the group.

`Input exponent law (0 if none):`

If you wish to construct only those immediate descendants which satisfy a particular exponent law, supply that exponent; if you do not wish to enforce an exponent law, supply `0`.

`Enforce metabelian law?`

If you answer "Yes", you seek to ensure that all of the immediate descendants constructed have the following property – if any one of them is used as a starting group to a later iteration, only the metabelian immediate descendants (if any) of this group are constructed. For this requirement to be enforceable, the starting group for this iteration must also have that property. To ensure that the starting group has that property, construct its p-covering group after having first set the metabelian flag via option `14` of the Advanced p-Quotient Menu (see 6.2).

`Do you want default output?`

If you answer "Yes", minimal output is displayed for each group: its identifier, the ranks of its p-multiplicator and nucleus, the number of its immediate descendants of each order, and, if there are any, the number of its capable immediate descendants.

For many applications, the default output obtained by "Yes" is sufficient. If not, then by answering "No" you are asked whether you want the default output for each of the following categories: permutation group, orbits, group descriptions, automorphism group descriptions, or if you want an algorithm trace. Answering "No" to any of these questions (except for the last) leads to further questions, each requiring a "Yes"/"No" answer, about what additional information you desire (or don't want) for the category. The following are the questions that result from a "No" response to "`Do you want default output?`":

`Do you want default permutation group output?`

A "No" leads to the further questions:

`Print degree of permutation group?`

`Print extended automorphisms?`

`Print automorphism matrices?`

`Print permutations?`

`Do you want default orbit output?`
A "No" leads to the further questions:
`Summary of orbit information?`
`Complete listing of orbits?`

`Do you want default group output?`
A "No" leads to the further questions:
`Print standard matrix of allowable subgroup?`
`Presentation of reduced p-covering groups?`
`Presentation of immediate descendants?`
`Print nuclear rank of descendants?`
`Print p-multiplicator rank of descendants?`

`Do you want default automorphism group output?`
A "No" leads to the further questions:
`Print commutator matrix?`
`Automorphism group description of descendants?`
`Automorphism group order of descendants?`

`Do you want algorithm trace output?`
This last question is designed to permit one to trace the intermediate stages of the algorithm.

After the above dialogue, the pq binary will commence constructing descendants. At the commencement of the application, each starting group is checked to determine whether it meets the selected step size order, and class criteria. If it does not, a message is displayed stating that this starting group is invalid.

7.6 Advanced Menu for p-Group Generation

```
Advanced Menu for p-Group Generation
------------------------------------
1. Read automorphism information for starting group
2. Extend and display automorphisms
3. Specify input file and group number
4. List group presentation
5. Carry out intermediate stage calculation
6. Compute definition sets & find degree
7. Construct permutations of subgroups under automorphisms
8. Compute and list orbit information
9. Process all orbit representatives
10. Process individual orbit representative
11. Compute label for standard matrix of subgroup
12. Compute standard matrix for subgroup from label
13. Find image of allowable subgroup under automorphism
14. Find rank of closure of initial segment subgroup
15. List representative and orbit for supplied label
16. Write compact descriptions of generated groups to file
17. Find automorphism classes of elements of vector space
18. Exit to main p-group generation menu
```

7.7 Strategies to minimise time and space

Where a PAG-generating sequence is supplied (i.e. the question "`PAG-generating sequence for automorphism group?`" is answered with a "Yes"; see 7.5), the minimum space requirement is achieved by supplying "`0`" and "Yes", respectively, to the questions: "`Rank of the initial segment subgroup?`" and "`Space efficient computation?`". This space efficiency is achieved at the cost of some additional time in computing the stabilisers of orbit representatives. However, if you simply wish to compute orbits, it is the best overall strategy, both from space and time considerations. The "efficient space" option is currently available only where a PAG-generating sequence is supplied.

In general, the most efficient time performance is obtained by taking the default algorithm (answering "Yes" to "`Default algorithm?`; see 7.5). This also gives significant space saving over most other strategies.

As mentioned earlier, the workspace size used in computing pcps – that is, the size of the array y – may be passed as a command line argument to the program at invocation. Much of the storage used in the implementation of p-group generation is separate from that allocated for y. Hence, if the program is to be used to generate group descriptions, it is probably sensible to invoke the program with a workspace size of no more than 100 000 rather than its default value, `PQSPACE` (which is defined in the header file, `constants.h`). See also the discussion on this point in the `README` file.

8 The Standard Presentation and Automorphism Group implementation

The Standard Presentation Menu allows a user to input a finite presentation for a group, to construct a "standard" presentation for a specified p-quotient of the group, and also to construct a description of the automorphism group of the p-group. To access this menu, you need to run the pq program with the `-i` run-time parameter switch (see 4).

The appropriate way to view the standard presentation algorithm is the following. The pcp constructed by supplying a finite presentation to the p-quotient algorithm depends on the supplied presentation. The standard presentation of a p-group obtained using the standard presentation algorithm is independent of the supplied presentation. Hence it allows us to determine whether two p-groups are isomorphic.

In its most general form, the "standard" presentation of a p-group is obtained by constructing a description of this group using the p-group generation algorithm.

The standard presentation of a class 1 p-quotient is identical to that obtained from the p-quotient. A user can choose to take the presentation returned from the p-quotient implementation to class k as an acceptable "standard" presentation up to that class and then proceed to standardise the presentation from class $k+1$ to some desired later class. This is particularly relevant if the user is seeking to verify that two groups are isomorphic. It may turn out that the two pcps constructed by the p-quotient implementation are identical up to class k. Since the standardisation procedure is significantly more expensive than a call to the p-quotient implementation, it makes sense in such situations to begin to standardise only from class $k+1$ onwards. However, the user must supply as input a description of the automorphism group of the class k

p-quotient – which may be more difficult to obtain for larger k.

In checking for isomorphism, it also makes sense to standardise each of the presentations, class by class. The standard presentations at the end of each class should be compared – if they are distinct, then the groups are non-isomorphic. In order to facilitate this, the program writes a file containing necessary details of the standard presentation and the automorphism group of the group up to the end of the specified class – this file can be used as input to the program later to continue the standardisation procedure. A generating set for a supplement to the inner automorphisms of the group is stored there; each generator is described by an $n \times n$ matrix whose exponents represent the image of each of the n pcp generators of the standard presentation.

```
Standard Presentation Menu
------------------------------
1. Supply start information
2. Compute standard presentation to supplied class
3. Save presentation to file
4. Display presentation
5. Set print level for construction
6. Compare two presentations stored in files
7. Call basic menu for p-Quotient program
8. Compute the isomorphism
9. Exit from program
```

We now describe each of these options.

`1. Supply start information`
: you must supply a finite presentation for the p-group; the queries are identical to that used in option `1` of the Basic Menu [for p-Quotient Program] (see 6.1). All of the valid formats for supplying a presentation can be accessed, using the `-b` or `-k` run-time switches (see Section 4). If the class supplied is c, then standardisation (selected under option `2`) begins at class $c + 1$ only. In general the supplied value for the class will be one – however, see the preceding discussion. A pcp for the class c p-quotient of the group is now computed using the the p-quotient implementation.

`2. Compute standard presentation to supplied class`
: If, when selecting this option, you haven't previously selected option `1` to supply a finite presentation, then you will be prompted:

  `Enter input file name for group information:`
  : The assumption is that such a file containing the presentation and automorphism information for the p-group was generated from a previous run of the Standard Presentation algorithm. If you don't supply a valid filename the pq program bails out of option `2`.

  Whether or not you have previously selected option `1`, you will then be prompted:

  `Enter output file name for group information:`
  : The file whose name you choose can be used as input later to continue the construction of the standard pcp. Then you will be asked:

`Standardise presentation to what class?`

The start class is one greater than the class of the p-quotient selected using option `1` or that stored on the input file. Here you should specify the end class for the standardisation procedure.

If you selected option `1` to supply a finite presentation, you will now be prompted for automorphism information – in exactly the same manner as under option `1` of the main Menu for p-Group Generation (see 7.4), and then also asked whether the supplied description is a PAG-generating sequence or not:

`PAG-generating sequence for automorphism group?`

`3. Save presentation to file`

`4. Display presentation`

print out the standard presentation to the current class.

`5. Set print level for construction`

ranges from 0 to 2. At print level 0, only timing information is printed. At print level 1, the standard presentation at the end of each class is also printed. At print level 2, full detail of the construction is reported. The default print level is 1.

`6. Compare two presentations stored in files`

supply the names of two data files containing these presentations; a check will be run to determine if the presentations are identical. This comparison facility may be applied to any two pcps – not just standard ones.

`7. Call basic menu for p-Quotient program`

provides access to the Basic Menu for p-Quotient program (see 6.1).

`8. Compute the isomorphism`

computes the mapping from the user-supplied generators to the generators for the standard presentation.

`9. Exit from program`

causes the pq program to exit. Selecting option `0` performs the same function.

Various files, all having prefixes "`ISOM_`", are first created and then deleted by pq while executing the standard presentation algorithm.

9 Warning

Pay attention to the results, and where possible confirm their correctness with other established sources.

A Examples

A.1 A Basic Menu example

The following example exercises options within the Basic Menu (see 6.1). When the pq program is used without any switches (see Section 4), it opens with the Basic Menu.

```
Basic Menu for p-Quotient Program
---------------------------------
1. Compute pc presentation
2. Save presentation to file
3. Restore presentation from file
4. Display presentation of group
5. Set print level
6. Calculate next class
7. Compute p-covering group
8. Advanced p-quotient menu
9. (Main) menu for p-group generation
10. Exit from p-quotient program

Select option: 1             #we want to enter a pc presentation
Input group identifier: 2gp  #something meaningful
Input prime: 2               #it's a 2-group
Input maximum class: 6
Input print level (0-3): 1   #minimal output
Input generating set (in { }): {a, b}
Input defining set of relations (in { }): { [b, a, a], (a * b * a)^4 }
Input exponent law (0 if none): 0

Lower exponent-2 central series for 2gp

Group: 2gp to lower exponent-2 central class 1 has order 2^2

Group: 2gp to lower exponent-2 central class 2 has order 2^5

Group: 2gp to lower exponent-2 central class 3 has order 2^8

Group: 2gp to lower exponent-2 central class 4 has order 2^11

Group: 2gp to lower exponent-2 central class 5 has order 2^15

Group: 2gp to lower exponent-2 central class 6 has order 2^19
Computation of presentation took 0.02 seconds

Select option: 2             #save option
Enter output file name: 2GP  #file name
Presentation written to file

Select option: 0             #exit
```

```
Exiting from ANU p-Quotient Program
Total user time in seconds is 0.02
```

This is essentially example `2gp` in the `examples` directory (except some of our #*comment*s are different). If the binary for the pq program is `pq`, then

```
pq < 2gp
```

executes the example non-interactively, with something similar to the above output to the screen, minus the menu. Note that the menus from the pq program are only displayed when it is used interactively. A script file for the pq program (like `2gp`) should contain the responses that the pq program will expect, in the correct sequence.

A.2 A p-Group Generation example

The following example is essentially `pga_3gp` from the `examples` directory, except again we have modified the comments, and we have also answered "No" to the "PAG-generating sequence" question, so that GAP is called to compute stabilisers. If the pq binary is unable to find GAP or finds GAP 3 instead of GAP 4 the program will die horribly at this point. (See Section 2 if you run into problemsm, at this point.)

For this example (which generates all groups with lower exponent-3 series of shape 2-2-3-1), the pq program is invoked without any of the switches of Section 4.

```
[..Basic Menu omitted here..]
Select option: 1         #set up group presentation
Input group identifier: c3c3
Input prime: 3
Input maximum class: 1
Input print level (0-3): 1
Input generating set (in { }): {a, b}
Input defining set of relations (in { }): {}
Input exponent law (0 if none): 0

Lower exponent-3 central series for c3c3

Group: c3c3 to lower exponent-3 central class 1 has order 3^2
Computation of presentation took 0.00 seconds

Select option: 7        #compute its 3-covering group

Group: c3c3 to lower exponent-3 central class 2 has order 3^5
Computation of 3-covering group took 0.00 seconds

Select option: 9        #enter p-group generation

Menu for p-Group Generation
---------------------------
1. Read automorphism information for starting group
2. Extend and display automorphisms
3. Specify input file and group number
```

```
4. List group presentation
5. Construct descendants
6. Advanced p-group generation menu
7. Exit to basic menu

Select option: 1                          #to supply automorphisms
Input the number of automorphisms: 5
Now enter the data for automorphism 1
Input 2 exponents for image of pcp generator 1: 2 0
Input 2 exponents for image of pcp generator 2: 0 2
Now enter the data for automorphism 2
Input 2 exponents for image of pcp generator 1: 0 2
Input 2 exponents for image of pcp generator 2: 1 0
Now enter the data for automorphism 3
Input 2 exponents for image of pcp generator 1: 1 2
Input 2 exponents for image of pcp generator 2: 2 2
Now enter the data for automorphism 4
Input 2 exponents for image of pcp generator 1: 1 0
Input 2 exponents for image of pcp generator 2: 2 1
Now enter the data for automorphism 5
Input 2 exponents for image of pcp generator 1: 2 0
Input 2 exponents for image of pcp generator 2: 0 1
Input number of soluble generators for automorphism group: 0

Select option: 5                          #to construct descendants
Input class bound on descendants: 4
Construct all descendants? 0              #i.e. ``No''
Constant step size? 0                     #i.e. ``No''
Input 3 step sizes: 2 3 1
PAG-generating sequence for automorphism group? 0 #i.e. ``No''
Do you want default algorithm? 1          #i.e. ``Yes''
Do you want default output? 1             #i.e. ``Yes''

************************************************
Starting group: c3c3
Order: 3^2
Nuclear rank: 3
3-multiplicator rank: 3
Now calling GAP to compute stabiliser...
true
#I  Order of GL subgroup is 48
#I  No. of soluble autos is 0
#I    dim U = 1  dim N = 3  dim M = 3
#I    nice stabilizer with perm rep
Now calling GAP to compute stabiliser...
true
#I  Order of GL subgroup is 48
#I  No. of soluble autos is 0
```

```
#I    dim U = 1  dim N = 3  dim M = 3
#I    nice stabilizer with perm rep
# of immediate descendants of order 3^4 is 3
# of capable immediate descendants is 3

*************************************************
3 capable groups saved on file c3c3_class2

*************************************************
Starting group: c3c3 #1;2
Order: 3^4
Nuclear rank: 2
3-multiplicator rank: 3
Group c3c3 #1;2 is an invalid starting group

*************************************************
Starting group: c3c3 #2;2
Order: 3^4
Nuclear rank: 3
3-multiplicator rank: 4
# of immediate descendants of order 3^7 is 4
# of capable immediate descendants is 4

*************************************************
Starting group: c3c3 #3;2
Order: 3^4
Nuclear rank: 2
3-multiplicator rank: 3
Group c3c3 #3;2 is an invalid starting group

*************************************************
4 capable groups saved on file c3c3_class3

*************************************************
Starting group: c3c3 #2;2 #1;3
Order: 3^7
Nuclear rank: 4
3-multiplicator rank: 5
# of immediate descendants of order 3^8 is 16
# of capable immediate descendants is 11

*************************************************
Starting group: c3c3 #2;2 #2;3
Order: 3^7
Nuclear rank: 3
3-multiplicator rank: 4
# of immediate descendants of order 3^8 is 13
# of capable immediate descendants is 9
```

```
************************************************
Starting group: c3c3 #2;2 #3;3
Order: 3^7
Nuclear rank: 3
3-multiplicator rank: 4
# of immediate descendants of order 3^8 is 13
# of capable immediate descendants is 9

************************************************
Starting group: c3c3 #2;2 #4;3
Order: 3^7
Nuclear rank: 3
3-multiplicator rank: 4
# of immediate descendants of order 3^8 is 7
# of capable immediate descendants is 5

************************************************
34 capable groups saved on file c3c3_class4
Construction of descendants took 69.95 seconds

Select option: 0           #exit to basic menu (same as option 7)
Exiting from p-group generation

Select option: 0           #exit program (same as option 9)
Exiting from ANU p-Quotient Program
Total user time in seconds is 69.95
```

A.3 A Standard Presentation Menu example

The following example is similar to what is provided by the file `2gp` in the `isom` directory, except we have added comments, we have not used the `-k` (keywords) runtime switch and to reduce the output we have only computed the standard presentation to class 3 (instead of class 10, as in the `2gp` example). For this example, the pq program is invoked with the `-i` runtime switch (see Section 4), which gives us access to the Standard Presentation Menu (see Section 8).

```
Standard Presentation Menu
---------------------------
1. Supply start information
2. Compute standard presentation to supplied class
3. Save presentation to file
4. Display presentation
5. Set print level for construction
6. Compare two presentations stored in files
7. Call p-Quotient menu
8. Compute the isomorphism
9. Exit from program
```

```
Select option: 1                    #input group info.
Input group identifier: G
Input prime: 2
Input maximum class: 1
Input print level (0-3): 1          #just minimal output
Input generating set (in { }): {a, b}
Input defining set of relations (in { }): {a^4, b^4 = [b, a, a]}
Input exponent law (0 if none): 0

Lower exponent-2 central series for G

Group: G to lower exponent-2 central class 1 has order 2^2
Class 1 2-quotient and its 2-covering group computed in 0.02 seconds

Select option: 2                    #for standard presentation
Enter output file name for group information: 2gp-st
Standardise presentation to what class? 3
Input the number of automorphisms: 2
Now enter the data for automorphism 1
Input 2 exponents for image of pcp generator 1: 0 1
Input 2 exponents for image of pcp generator 2: 1 1
Now enter the data for automorphism 2
Input 2 exponents for image of pcp generator 1: 0 1
Input 2 exponents for image of pcp generator 2: 1 0
PAG-generating sequence for automorphism group? 1  #i.e. ``Yes''
Starting group has order 2^2; its automorphism group order is 6

The standard presentation for the class 2 2-quotient is

Group: G #1;3 to lower exponent-2 central class 2 has order 2^5

Non-trivial powers:
 .1^2 = .4
 .2^2 = .5

Non-trivial commutators:
[ .2, .1 ] = .3
Its automorphism group has order 384
Computing standard presentation for class 2 took 0.08 seconds

The standard presentation for the class 3 2-quotient is

Group: G #1;3 to lower exponent-2 central class 3 has order 2^8

Non-trivial powers:
 .1^2 = .4
 .2^2 = .5
```

```
 .3^2 = .6 .8
 .5^2 = .6

Non-trivial commutators:
[ .2, .1 ] = .3
[ .3, .1 ] = .6
[ .3, .2 ] = .7
[ .4, .2 ] = .8
[ .5, .1 ] = .6 .7 .8
Its automorphism group has order 4096
Computing standard presentation for class 3 took 0.12 seconds

Select option: 0           #exit (option 9 does this also)
```

A.4 A keywords example

To use keywords you must use the `-k` runtime switch (see Section 4). For this example we are again using the Standard Presentation Menu; so we also use the `-i` runtime switch. The example is `2gp.com` from the `isom` directory. If the binary for the pq program is `pq`, then

```
pq -i -k < 2gp.com
```

in the `isom` directory yields something like the following output. (The keywords are used in option 1; note the "`;`" indicating all remaining data for that option should take default values.)

```
[..Standard Presentation Menu omitted here..]

Select option: 1          #set up start information
prime 2                   #keyword 'prime'
class 2                   #keyword 'class'
generators {a, b}         #keyword 'generators'
relations  {a^4, b^2 = [b, a, b]}; #keyword 'relations' (NB: closing ';')

Lower exponent-2 central series for G

Group: G to lower exponent-2 central class 1 has order 2^2

Group: G to lower exponent-2 central class 2 has order 2^4
Class 2 2-quotient and its 2-covering group computed in 0.00 seconds

Select option: 2          #standardise presentation
Enter output file name for group information: Standard
Standardise presentation to what class? 10
Input the number of automorphisms: 3
Now enter the data for automorphism 1
Input 4 exponents for image of pcp generator 1: 1 0 0 1
Input 4 exponents for image of pcp generator 2: 0 1 0 0
```

```
Now enter the data for automorphism 2
Input 4 exponents for image of pcp generator 1: 1 0 0 0
Input 4 exponents for image of pcp generator 2: 0 1 0 1
Now enter the data for automorphism 3
Input 4 exponents for image of pcp generator 1: 1 1 1 0
Input 4 exponents for image of pcp generator 2: 0 1 1 1
PAG-generating sequence for automorphism group? 1
Starting group has order 2^4; its automorphism group order is at most 96

The standard presentation for the class 3 2-quotient is

Group: G #1;2 to lower exponent-2 central class 3 has order 2^6

Non-trivial powers:
 .1^2 = .4
 .2^2 = .5
 .3^2 = .6

Non-trivial commutators:
[ .2, .1 ] = .3
[ .3, .1 ] = .5
[ .3, .2 ] = .6
[ .4, .2 ] = .5 .6
Its automorphism group has order at most 384
Computing standard presentation for class 3 took 0.05 seconds

The standard presentation for the class 4 2-quotient is

Group: G #1;1 to lower exponent-2 central class 4 has order 2^7
[..265 lines omitted here..]

The standard presentation for the class 10 2-quotient is

Group: G #1;4 to lower exponent-2 central class 10 has order 2^24
[..93 lines omitted here..]

Its automorphism group has order at most 25769803776
Computing standard presentation for class 10 took 0.27 seconds

Select option: 4 #display standard presentation for class 10 2-quotient

Group: G #1;4 to lower exponent-2 central class 10 has order 2^24

Non-trivial powers:
 .1^2 = .4
 .2^2 = .5 .11 .13 .17 .24
 .3^2 = .6 .7 .10 .11 .12 .13 .14 .17 .20 .21 .22
 .5^2 = .8 .11 .19 .20 .21
```

```
 .6^2 = .10 .14 .16 .18 .22 .23 .24
 .7^2 = .10 .12 .21 .22 .24
 .8^2 = .12 .15
 .9^2 = .12 .17 .20 .22 .23
 .10^2 = .15 .18
 .11^2 = .15 .20 .24
 .12^2 = .18 .21
 .13^2 = .18 .24
 .14^2 = .18 .23
 .15^2 = .21
 .17^2 = .21

Non-trivial commutators:
[ .2, .1 ] = .3
[ .3, .1 ] = .5
[ .3, .2 ] = .6
[..71 lines omitted here..]
[ .20, .2 ] = .23

Select option: 0     #exit
Exiting from ANU p-Quotient Program
Total user time in seconds is 1.13
```

A.5 A basic format Advanced p-Group Generation example

With the following example we demonstrate both the use of the `-b` runtime switch (see Section 4) and we exercise an Advanced menu. To do something equivalent to what we give below, (assuming the pq binary is `pq`) do:

```
pq -b < pga_interactive
```

in the `examples` directory.

```
[..Basic Menu omitted here..]
Select option: 1               #set up group presentation
Input group identifier: Nott   #our group will be the Nottingham group
Input prime: 5
Input maximum class: 3
Input print level (0-3): 1     #minimal output
Input number of generators: 2  #this is the prompt we get with -b
Input number of relations: 3   #(ditto)
Input exponent law (0 if none): 0
5 1;                           #this is the ‘‘basic format’’ (-b switch)
The input word is 5 1
Input right-hand side of relation:
0;
The input word is 0
Input left-hand side of relation:
5 2;
```

```
The input word is 5 2
Input right-hand side of relation:
0;
The input word is 0
Input left-hand side of relation:
1 [2 1 2];
The input word is 1 [ 2 1 2 ]
Input right-hand side of relation:
0;                              #the final ‘‘basic format’’ input needed
The input word is 0

Lower exponent-5 central series for Nott

Group: Nott to lower exponent-5 central class 1 has order 5^2

Group: Nott to lower exponent-5 central class 2 has order 5^3

Group: Nott to lower exponent-5 central class 3 has order 5^4
Computation of presentation took 0.00 seconds

Select option: 7                #compute 5-covering group

Group: Nott to lower exponent-5 central class 4 has order 5^8
Computation of 5-covering group took 0.00 seconds

Select option: 2                #save presentation
Enter output file name: Nott
Presentation written to file

Select option: 9                #to (Main) p-Group Generation Menu

Menu for p-Group Generation
---------------------------
1. Read automorphism information for starting group
2. Extend and display automorphisms
3. Specify input file and group number
4. List group presentation
5. Construct descendants
6. Advanced p-group generation menu
7. Exit to basic menu

Select option: 1                #define aut. group
Input the number of automorphisms: 6
Now enter the data for automorphism 1
Input 4 exponents for image of pcp generator 1: 1 0 0 0
Input 4 exponents for image of pcp generator 2: 0 1 0 1
Now enter the data for automorphism 2
Input 4 exponents for image of pcp generator 1: 1 1 0 0
```

```
Input 4 exponents for image of pcp generator 2: 0 1 0 0
Now enter the data for automorphism 3
Input 4 exponents for image of pcp generator 1: 1 0 0 0
Input 4 exponents for image of pcp generator 2: 0 4 0 0
Now enter the data for automorphism 4
Input 4 exponents for image of pcp generator 1: 1 0 0 0
Input 4 exponents for image of pcp generator 2: 0 2 0 0
Now enter the data for automorphism 5
Input 4 exponents for image of pcp generator 1: 4 0 0 0
Input 4 exponents for image of pcp generator 2: 0 1 0 0
Now enter the data for automorphism 6
Input 4 exponents for image of pcp generator 1: 2 0 0 0
Input 4 exponents for image of pcp generator 2: 0 1 0 0
Input number of soluble generators for automorphism group: 0

Select option: 5              #construct descendants
Input class bound on descendants: 4
Construct all descendants? 0   #i.e. ``No''
Input step size: 1
PAG-generating sequence for automorphism group? 1 #``Yes''
Do you want default algorithm? 0                  #``No''
Rank of the initial segment subgroup? 4
Space efficient computation? 0                    #``No''
Completely process terminal descendants? 0        #``No''
Input exponent law (0 if none): 0
Enforce metabelian law? 0                         #``No''
Do you want default output? 1                     #``Yes''

************************************************
Starting group: Nott
Order: 5^4
Nuclear rank: 1
5-multiplicator rank: 4
# of immediate descendants of order 5^5 is 9
# of capable immediate descendants is 2

************************************************
2 capable groups saved on file Nott_class4
Construction of descendants took 0.02 seconds

Select option: 3              #restore group
Enter input file name: Nott_class4
Which group? 1

Select option: 6              #get Advanced p-Group Gen'n Menu

Advanced Menu for p-Group Generation
------------------------------------
```

```
1. Read automorphism information for starting group
2. Extend and display automorphisms
3. Specify input file and group number
4. List group presentation
5. Carry out intermediate stage calculation
6. Compute definition sets & find degree
7. Construct permutations of subgroups under automorphisms
8. Compute and list orbit information
9. Process all orbit representatives
10. Process individual orbit representative
11. Compute label for standard matrix of subgroup
12. Compute standard matrix for subgroup from label
13. Find image of allowable subgroup under automorphism
14. Find rank of closure of initial segment subgroup
15. List representative and orbit for supplied label
16. Write compact descriptions of generated groups to file
17. Find automorphism classes of elements of vector space
18. Exit to main p-group generation menu

Select option: 6              #find degree
Input step size: 2
Rank of the initial segment subgroup? 3
Input exponent law (0 if none): 0
Degree of permutation group is 25

Select option: 7              #compute permutations
PAG-generating sequence for automorphism group? 1
Space efficient computation? 0
Print automorphism matrices? 0
Print permutations? 0
Time to compute permutations is 0.00 seconds

Select option: 8              #compute orbits
PAG-generating sequence for automorphism group? 1 #``Yes''
Space efficient computation? 0                    #``No''
Summary of orbit information? 1                   #``Yes''
Complete listing of orbits? 0                     #``No''
Time to compute orbits is 0.00 seconds

  Orbit          Length      Representative
      1               5               1
      2              20               2


Select option: 9              #compute stabilisers
PAG-generating sequence for automorphism group? 1 #``Yes''
Space efficient computation? 0                    #``No''
Completely process terminal descendants? 0        #``No''
```

```
Input exponent law (0 if none): 0
Enforce metabelian law? 0                          #``No''
Print standard matrix of allowable subgroup? 0     #``No''
Presentation of reduced p-covering groups? 0       #``No''
Presentation of immediate descendants? 0           #``No''
Print nuclear rank of descendants? 0               #``No''
Print p-multiplicator rank of descendants? 0       #``No''
Print commutator matrix? 0                         #``No''
Automorphism group description of descendants? 0   #``No''
Automorphism group order of descendants? 0         #``No''
Enter output file name: X
Time to process representative is 0.02 seconds

Select option: 3               #restore reduced p-covering group
Enter input file name: X
Which group? 1

Select option: 4               #display presentation

Group: Nott #1;1 to lower exponent-5 central class 5 has order 5^9
Class 1
1 is defined on image of defining generator 1
2 is defined on image of defining generator 2
Class 2
3 is defined on [2, 1] = 2 1
Class 3
4 is defined on [3, 1] = 2 1 1
Class 4
5 is defined on [4, 1] = 2 1 1 1
Class 5
6 is defined on [5, 1] = 2 1 1 1 1
7 is defined on [5, 2] = 2 1 1 1 2
8 is defined on 1^5 = 1 1
9 is defined on 2^5 = 2 2

Non-trivial powers:
 .1^5 = .8
 .2^5 = .9

Non-trivial commutators:
[ .2, .1 ] = .3
[ .3, .1 ] = .4
[ .4, .1 ] = .5
[ .4, .2 ] = .7
[ .4, .3 ] = .7^4
[ .5, .1 ] = .6
[ .5, .2 ] = .7
```

```
Select option: 5              #intermediate stage computation
Input step size: 2
PAG-generating sequence for automorphism group? 1 #``Yes''
Do you want default algorithm? 1                  #``Yes''
Do you want default output? 1                     #``Yes''
Input output file name: XX

************************************************
Starting group: Nott #1;1
Order: 5^5
Nuclear rank: 2
5-multiplicator rank: 4
# of immediate descendants of order 5^7 is 40
# of capable immediate descendants is 5
Time for intermediate stage is 0.07 seconds

Select option: 0              #exit through 3 levels of menus
Exiting from advanced p-group generation menu

Select option: 0
Exiting from p-group generation

Select option: 0
Exiting from ANU p-Quotient Program
Total user time in seconds is 0.10
```

B Changes

`Version 1.9`
: Fixes made to warnings in C code by Max Horn.

`Version 1.8`
: GAP link code updated for GAP 4.4.

`Version 1.7`
: Binomial coefficient algorithm modified to avoid overflow.

`Version 1.6`
: In main Menu for p-Group Generation, when asking for automorphisms, now ask for the number of soluble automorphisms and their relative orders when there are any.

`Version 1.5`
: Added the `-G` option for use with GAP and the `-v` option.

References

George Havas and M.F. Newman (1980), "Application of computers to questions like those of Burnside", *Burnside Groups* (Bielefeld, 1977), *Lecture Notes in Math.* **806**, pp. 211-230. Springer-Verlag, Berlin, Heidelberg, New York.

M.F. Newman (1977), "Determination of groups of prime-power order", *Group Theory* (Canberra, 1975). *Lecture Notes in Math.* **573**, pp. 73–84. Springer-Verlag.

M.F. Newman and E.A. O'Brien (1996), "Application of computers to questions like those of Burnside, II", *Internat. J. Algebra Comput.* **6**, 593-605.

E.A. O'Brien (1990), "The p-group generation algorithm", *J. Symbolic Comput.* **9**, 677-698.

E.A. O'Brien (1994), "Isomorphism testing for p-groups", *J. Symbolic Comput.* **17**, 133–147.

E.A. O'Brien (1995), "Computing automorphism groups for p-groups", *Computational Algebra and Number Theory* (Sydney, 1992), pp. 83–90. Kluwer Academic Publishers, Dordrecht.

M.R. Vaughan-Lee (1982), "An Aspect of the Nilpotent Quotient Algorithm", *Computational Group Theory* (Durham, 1982), pp. 76–83. Academic Press.

Michael Vaughan-Lee (1990a), "The Restricted Burnside Problem", *London Mathematical Society monographs (New Ser.)* **5**. Clarendon Press, New York, Oxford.

M.R. Vaughan-Lee (1990b), "Collection from the left", *J. Symbolic Comput.* **9**, 725–733.

Eamonn A. O'Brien
Department of Mathematics
University of Auckland
Private Bag 92019, Auckland, New Zealand

E-mail address: obrien@math.auckland.ac.nz

Last revised by Eamonn O'Brien: August 2001
Revised February 2002 (v1.5 GG), February 2004 (v1.6 GG),
November 2011 (v1.9 MH), January 2012 (v1.9 GG)