Neper Reference Manual

The documentation for Neper 1.10.0
A 3D random polycrystal generator for the finite element method

4 June 2012

Romain Quey

Table of Contents

Conditions of Use

Copying Conditions

Neper is "free software"; this means that everyone is free to use it and to redistribute it on a free basis. Neper is not in the public domain; it is copyrighted and there are restrictions on its distribution, but these restrictions are designed to permit everything that a good cooperating citizen would want to do. What is not allowed is to try to prevent others from further sharing any version of Neper that they might get from you.

Specifically, we want to make sure that you have the right to give away copies of Neper, that you receive source code or else can get it if you want it, that you can change Neper or use pieces of Neper in new free programs, and that you know you can do these things.

To make sure that everyone has such rights, we have to forbid you to deprive anyone else of these rights. For example, if you distribute copies of Neper, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must tell them their rights.

Also, for our own protection, we must make certain that everyone finds out that there is no warranty for Neper. If Neper is modified by someone else and passed on, we want their recipients to know that what they have is not what we distributed, so that any problems introduced by others will not reflect on our reputation.

The precise conditions of the license for Neper are found in the General Public License that accompanies the source code (see Appendix E [GNU General Public License], page 53). Further information about this license is available from the GNU Project webpage `http://www.gnu.org/copyleft/gpl-faq.html`.

The Neper software package can be downloaded from `http://neper.sourceforge.net`. It has also two dedicated mailing lists,

- neper-announce: the "read-only" list for important news: new releases, bug fixes, etc. (low traffic, highly recommended!)

  To subscribe, visit `https://lists.sourceforge.net/lists/listinfo/neper-announce`. The list is archived at `http://sourceforge.net/mailarchive/forum.php?forum_name=neper-announce`.
- neper-users: the "read-write" list for users. Please send all questions, bug reports, requests or any errors or omissions in this manual to this list.

  To subscribe, visit `https://lists.sourceforge.net/lists/listinfo/neper-users`; to send a message, use `neper-users@lists.sourceforge.net`. The list is archived at `http://sourceforge.net/mailarchive/forum.php?forum_name=neper-users`.

The best way to get help is by sending a message to `neper-users@lists.sourceforge.net`. Note that you do not have to subscribe to the list to be able to send a message, nor to receive the reply. If you wish to contact the developer directly, use `rquey@users.sourceforge.net`.

User Guidelines

If you use Neper for your own work, please cite it in your reports (books, papers, talks, . . .). The Neper references are given below (if you do not wish to cite both, please cite the first one).

- *R. Quey, P.R. Dawson, F. Barbe. Large-scale 3D random polycrystals for the finite element method: Generation, meshing and remeshing. Computer Methods in Applied Mechanics and Engineering, vol. 200, pp. 1729–1745, 2011.*
- *Neper: a 3D random polycrystal generator for the finite element method (version 1.10),* `http://neper.sourceforge.net`.

1 Introduction

1.1 The Neper Project

1.1.1 Description

Neper is a 3D random polycrystal generator for the finite element method. It is built around several modules:

- Module -T is for generating polycrystal morphologies. They are described as space-filling tessellations of space whose vertices, edges, faces and volumes, represent the quadruple points, triple lines, grain boundaries and grains of the polycrystals, respectively. The polycrystal morphologies can be random Voronoi tessellations, or regular tessellations made of truncated octahedra. The tessellations are brick-shape by default, but can be of any convex shape. After they have been generated, they can be deformed to account for morphological texture.
- Module -FM aims at generating free (or "unstructured") meshes of tessellations, that is, meshes comprised of tetrahedral elements that conform to the tessellation morphology. Neper includes several advanced features that are necessary to get good-quality meshes: optimized meshing rules, a geometry regularization approach, multimeshing (the competitive use of several meshers) and remeshing.
- Module -MM is for generating mapped meshes of tessellations. These meshes are comprised of regular hexahedral elements and so do not conform exactly to the tessellation morphology. Mapped meshes of standard tessellations, periodic tessellations and herein called *subdomain-type* tessellations can be created.
- Module -O provides crystallographic orientations for the grains. The orientations are randomly distributed according to a uniform distribution.
- Module -VS is for printing publication-quality images of the tessellations and meshes.

Neper aims to be an easy-to-use, efficient and robust tool. All the input data are prescribed non-interactively, using command lines and/or ASCII files. This makes it possible to automate all treatments.

1.1.2 Resources

Several, complementary resources describing the Neper capabilities are available:

- The Neper reference manual. It describes all the Neper capabilities. Each module is the subject of a specific chapter, in which all the available commands as well as the result files are described in detail. Some examples are also provided. The Neper documentation comes in two formats: a PDF file and an info file. Provided that the info file is properly installed at your site, it can be accessed by running the shell command: `info neper`.
- The Neper homepage: `http://neper.sourceforge.net`. It is where the Neper distribution can be downloaded from. It also provides an introduction to Neper, with some examples and illustrations.
- The Neper paper, "*R. Quey, P.R. Dawson and F. Barbe, Large-scale 3D random polycrystals for the finite element method: Generation, meshing and remeshing, Comput. Methods Appl. Mech. Engrg., vol. 200, pp. 1729-1745, 2011.*", provides details on the algorithms. It can be downloaded from the Neper homepage or by following this link: `http://neper.sourceforge.net/neper_paper.pdf`.

1.2 Installing Neper

Neper is written in (mostly ANSI) C and can run on any Unix-like system. Neper can be compiled using Cmake, a standard open-source build system. The main steps are as follows,

- Create a '`build`' directory, for example as a subdirectory of Neper's '`src`' directory,

  `$ mkdir build`

- Run cmake from within the '`build`' directory, pointing to Neper's '`src`' directory,

  `$ cd build`

  `$ cmake ..`

- To build and install Neper, then simply type,

  `$ make`

  `$ make install (as root)`

This will use the default configuration options, including on which dependencies are used, and should work out of the box provided that the required libraries are installed in standard system locations. A finer configuration can be achieved before building Neper, as described in the following.

1.2.1 Dependencies

Neper has mandatory as well as optional dependencies. Some dependencies are needed at compilation time. They are,

- the GSL library (mandatory)

  It is likely to be available on your system or from your system package manager (packages `gsl` and `gsl-devel`). Alternatively, the source code version can be obtained from the GSL homepage, `http://www.gnu.org/software/gsl`.

- the libmatheval library (optional, *included* by default)

  It is likely to be available on your system or from your system package manager (packages `libmatheval` and `libmatheval-devel`). Alternatively, the source code version can be obtained from the libmatheval homepage, `http://www.gnu.org/software/libmatheval`.

- the libScotch library (optional, *not included* by default)

  Module -FM includes mesh partitioning capabilities, which make use of the Scotch mesh partitioner (version `5.1.12` or later). It can be downloaded from the Scotch homepage, `www.labri.fr/perso/pelegrin/scotch`.

Optional dependencies can be toggled on / off using Cmake's GUI (`cmake-gui`) or `ccmake`, by setting variables `HAVE_`*`LIBRARYNAME`* to ON or OFF, respectively.

The following dependencies are not managed during the compilation. They are only needed at run time,

- the Gmsh program (mandatory for module -FM)

  This version on Neper is intended to work with Gmsh (version `2.4.2` or later), which can be downloaded from `http://www.geuz.org/gmsh`. A working Gmsh installation must be available on your system.

- the POV-Ray program (mandatory for module -VS)

  Module -VS uses POV-Ray to produce publication-quality images of the tessellations and meshes. POV-Ray can be downloaded from `http://www.povray.org`. POV-Ray must be available in the terminal through the command: `povray`.

1.3 Getting Started

Using Neper consists in running the command '`neper`' in a terminal, with a list of arguments,

```
$ neper list_of_arguments
```

The arguments define the problem for Neper to solve. Neper then returns output in ASCII files, together with some messages in the terminal. Neper includes some general-purpose self-explanatory commands,

```
$ neper --help
$ neper --version
$ neper --license
```

1.3.1 Call a Module

A typical Neper invocation consists in calling a module and providing it with a number of arguments:

```
$ neper module_name module_arguments
```

The module names are '`-T`', '`-FM`', '`-MM`', '`-O`' and '`-VS`'. The module arguments can include both required input data and options. Options start by '`-`'. The options can be given in arbitrary order (except for module `-VS`) and are to be specified as follows: "*option_name option_value*". The options can be written both in British English and in American English, although only the British English versions are indicated in this manual. String completion is available for all arguments, so they may be abbreviated as long as the abbreviation is not ambiguous. For example, in module `-O`, the option '`-descriptor`' can be abbreviated to '`-des`' or even safely to '`-d`'. Logical options can be selected by giving the value '`1`' or disabled by giving '`0`'. Neper is highly parametrable, and as a consequence includes quite a large number of options. For clarity sake, they are tagged according to their importance level in the reference manual: '[Option]' or '[Secondary option]'. The post-processing options are tagged '[Post-processing]'.

1.3.2 Initialization File

When Neper is started, it reads commands from an initialization file, '`$HOME/.neperrc`', if that file exists. This behaviour can be modified through option '`--rcfile`', which has to be loaded *prior to* calling a module,

```
$ neper --rcfile my_file module_name module_arguments
```

To inhibit the reading of an initialization file, provide '`none`' as value of the *my_file* argument.

When a module of Neper is called, Neper looks for the occurrence of '`neper` *`module_name`*' in the initialization file, then reads all the arguments until the subsequent occurrence of '`neper`' (which should denote the beginning of another module option field) or the end of the file. Moreover, any comments can be written after giving '`neper comments`'. The arguments may be any legal arguments, but are typically limited to frequently-used options.

An example of initialization file is given below:

```
neper comments -------------------------------------------------------
This is my default initialization file (~/.neperrc).
neper -FM -order 2 -maxff 20 -gmsh my_gmsh_path
          -rcl 0.8
neper -MM -order 2
neper comments -------------------------------------------------------
```

If '`neper` *`module_name`*' is not found, or if the initialization file is not found, Neper will just consider the command line arguments. Also, note that if an argument is initialized several times (for example, both in the initialization file and at the command line), the last specified value is retained.

1.3.3 Conventions

1.3.3.1 Manual

The Neper documentation is maintained as a Texinfo manual. Here are the writing conventions used in the document:

1. A command that can be typed in a terminal is printed like `this`, or, in the case of a major command, like

   ```
   $ this
   ```

2. a program (or command) option is printed like '`this`';
3. The name of a variable is printed like `this`;
4. A metasyntactic variable (i.e. something that stands for another piece of text) is printed like *this*;
5. Literal examples are printed like '`this`';
6. File names are printed like '`this`'.

1.3.3.2 Option Arguments

When using the different Neper modules, you may want to provide several values to a given option. A typical example is when you want to set the format of the tessellation or mesh files. In this case, the several values can be specified at the same time, separated by commas. Neper will process them sequentially. For instance, in module -T, you may use option '`-format tess,ply`' to get the tessellation both in Neper's tessellation format '`.tess`' and in Ply format '`.ply`'.

2 Tessellation Generation: neper -T

Module -T enables one to generate Voronoi tessellations of a space *domain*. The domain can be cuboidal, cylindrical or of any other convex shape. The centres of the polyhedra can be randomly distributed in the domain, which leads to *random Voronoi tessellations* (also called *Poisson Voronoi tessellations*). It is also possible to generate columnar (2D) and bamboo (1D) grain morphologies, as well as regular, periodic, arrangment of polyhedra (truncated octahedra are currently available). It is also possible to generate centroidal Voronoi tessellations[1], or even to load a user-defined distribution of centres. The tessellations can be scaled to generate morphological textures (flat or elongated grains). The module generates as output a tessellation file '`.tess`' that describes exhaustively the polycrystal morphology (see Appendix A [File Formats], page 43). The '`.tess`' file is an input file of the meshing modules, -FM and -MM (see Chapter 3 [Module -FM], page 13 and Chapter 4 [Module -MM], page 23). Module -T also generates as output a '`.oin`' file, which contains input data for the crystallographic orientation generation with module -O (see Chapter 5 [Module -O], page 27). The generated tessellation ('`.tess`' file) can be visualized with module -VS (Chapter 6 [Module -VS], page 31).

Here is what a typical run of module -T looks like:

```
$ neper -T -n 10 -id 1

========================    N   e   p   e   r    =======================
Info   : A 3D random polycrystal generator for the finite element method
Info   : Version 1.10.0
Info   : (compiled with: gsl libmatheval)
Info   : Loading initialization file '/home/rquey/.neperrc'...
Info   : ---------------------------------------------------------------
Info   : MODULE  -T loaded with arguments:
Info   : [ini file] (none)
Info   : [com line] -n 10 -id 1
Info   : ---------------------------------------------------------------
Info   : Reading input data ...
Info   : Creating domain ...
Info   : Creating polyhedron centres ...
Info   : Creating tessellation ...
Info   : Writing results ...
Info   :     [o] Writing file 'n10-id1.tess' ...
Info   :     [o] Wrote file 'n10-id1.tess'.
Info   :     [o] Writing file 'n10-id1.oin' ...
Info   :     [o] Wrote file 'n10-id1.oin'.
Info   : Elapsed time: 0.004 secs.
========================================================================
```

[1] Q. Du, V. Faber and M. Gunzburger, Centroidal Voronoi Tessellations: Applications and Algorithms, SIAM Review, 41, 637–676, 1999.

2.1 Arguments

2.1.1 Input Data

By default, the domain to tessellate is taken as a cube of unit size. For defining another domain, see option '`-domain`'.

`-n` *`integer`* [Input data]
Number of polyhedra of the tessellation, except for regular morphologies (truncated octahedra, etc., see option '`-morpho`' for details).
Possible values: `any`. Default value: `none`.

`-id` *`integer`* [Input data]
Identifier of the tessellation.
Possible values: `any`. Default value: *`random`*.

`-morpho` *`char_string`* [Input data]
Type of morphology of the polyhedra. For random Voronoi tessellations, it can be either equiaxed (`equiaxed`), columnar (`columnar{x,y,z}`) or bamboo-like (`bamboo{x,y,z}`).
To load a particular set of polyhedron centres, use the syntax '`@`*`file_name`*' where *`file_name`* is the name of a file containing the $3*n$ coordinates.
Regular morphologies can also be obtained: truncated octahedra (`tocta`), in which case the value of option '`-n`' stands for the number of polyhedra along an edge of the domain instead of the total number of polyhedra.
Possible values: `see above list`. Default value: `equiaxed`.

`-centroid` *`logical`* [Input data]
Use this option to turn the tessellation centroidal.
Possible values: `0 or 1`. Default value: `0`.

The generation of a centroidal Voronoi tessellation is based on Lloyd's algorithm and can be two to three orders of magnitude as long as for the equivalent Voronoi tessellation. The convergence criteria can be adjusted using options '`-centroidfact`', '`-centroidconv`' and '`-centroiditermax`'.

Is it also possible to load a tessellation from a file,

`-loadtess` *`file_name`* [Input data]
Load a tessellation from a file. Provide as argument the file name.
Possible values: `any`. Default value: `none`.

2.1.2 General Options

`-o` *`file_name`* [Option]
Specify the output file name.
Possible values: `any`. Default value: `none`.

2.1.3 Domain Options

`-domain` *`char_string ...`* [Option]
Specify the type of domain and its size. For a cuboidal shape, provide '`cube`' followed by the sizes along the x, y and z directions. For a cylindrical shape, provide '`cylinder`' followed by the height and diameter. For a tessellation polyhedron, provide '`tesspoly`' followed by the name of the tessellation file ('`.tess`') and the number of the polyhedron (the tessellation must not be regularized). For an arbitrary shape, provide '`planes`' followed by the name of a file containing the total number of planes then, for each plane, the parameters of its equation

(a, b, c and d for an equation of the form $a\,x + b\,y + c\,z = d$). The plane normal must be an outgoing vector of the polyhedron.
Possible values: `see above list`. Default value: `cube 1 1 1`.

`-cylinderfacet` *`integer`* [Secondary option]
Specify the number of facets along the circular part of a cylindrical domain.
Possible values: `any >= 3`. Default value: `function of the poly rad`.

2.1.4 Tessellation Options

`-scale` *`real real real`* [Option]
Specify the factors in the x, y and z directions by which the tessellation is to be scaled (once generated).
Possible values: `any`. Default value: `none`.

`-sort` *`char_string char_string`* [Secondary option]
This option can be used to sort the tessellation entities (typically to facilitate data post-processing). The first argument is the type of entity to sort (must be `poly`) and the second argument is the mathematical expression used for sorting (see Appendix B [Mathematical Expressions], page 47). For polyhedra, the available variables are the centre coordinates, *cenx*, *ceny* and *cenz*, the true and body parameters *true* and *body*, and the volume *vol*.
Possible values: `any`. Default value: `none`.

`-randomize` *`real integer`* [Secondary option]
This option can be used to "randomize" the coordinates of the polyhedron centres. Provide as argument the maximum shift distance and an identifier for the randomization. The resulting centre shift distances are uniformly distributed between 0 and the maximum shift distance.
Possible values: `any > 0 any`. Default value: `none`.

`-randomizedir` *`char_string`* [Secondary option]
This option can be used with option '`-randomize`', to specify the possible directions in which the polyhedron centres are to be shifted. Provide as argument the directions. Combine them with '`,`'.
Possible values: `x, y, z`. Default value: `x,y,z`.

`-centroidfact` *`real`* [Secondary option]
This option can be used with option '`-morpho centroid`', to specify the factor by which the germ positions are shifted between their current positions and the centroid positions, at each iteration. (Lloyd's algorithm is obtained for a value of 1, but a lower value can lead to faster convergence.)
Possible values: `0 to 1`. Default value: `0.5`.

`-centroidconv` *`real`* [Secondary option]
This option can be used with option '`-morpho centroid`', to specify the maximum tolerance on the distance between the polyhedron centres and centroids. The tolerance is relative to the average grain radius.
Possible values: `any > 0`. Default value: `0.02`.

`-centroiditermax` *`integer`* [Secondary option]
This option can be used with option '`-morpho centroid`', to specify the maximum number of iterations. You should consider using option '`-centroidconv`' instead.
Possible values: `any`. Default value: `1000`.

2.1.5 Output Options

`-format` *`char_string`* [Option]

Specify the format of the output file(s). The available formats are the Neper `tess` and `oin`, the Gmsh `geo` and the Ply `ply` (combine with commas).
Possible values: `tess, oin, geo, ply`. Default value: `tess,oin`.

2.1.6 Post-Processing Options

`-stat` *`logical`* [Post-processing]

Provide statistics on the tessellation.
Possible values: `0 or 1`. Default value: `0`.
Result file: extension '`.stt#`'.

`-neighbour` *`logical`* [Post-processing]

Provide a file with information on the first neighbours of the polyhedra.
Possible values: `any`. Default value: `none`.
Result file: extension '`.neigh`'.

`-pointpoly` *`file_name`* [Post-processing]

Provide the numbers of the polyhedra of which specific points belong. Give as argument the name of the file containing the coordinates of the points.
Possible values: `any`. Default value: `none`.
Result file: extension '`.polyid`'.

2.2 Output Files

2.2.1 Tessellation

- Neper-native tessellation file: '`.tess`'

  It contains an exhaustive description of the tessellation. See Appendix A [File Formats], page 43 for the file syntax.
- Gmsh geometry file: '`.geo`'

  It contains a minimal description of the tessellation written under the Gmsh geometry file format '`.geo`'. This file can be opened with Gmsh for visualization. (Note that you can even get a mesh of the tessellation out from Gmsh, but it will be of lower quality than by using module `-FM`.)
- Ply file: '`.ply`'

  It contains a description of the tessellation written under the standard "Polygon File Format" '`.ply`'.

2.2.2 Orientation

- orientation input file: '`.oin`'

  It contains data for generating the grain orientations, and is an input file for module -O (see Chapter 5 [Module -O], page 27).

2.2.3 Statistics

Several files are provided for statistics on tessellations, whose formats are provided below. All files are formatted with one entity (vertex, edge, face or polyhedron) per line.

- tessellation vertex statistics, '`.stt0`': *`id true body state x y z`*
- tessellation edge statistics, '`.stt1`': *`id true body state length`*

- tessellation face statistics, '`.stt2`': *`id true body state ver_qty area ff`*
- tessellation polyhedron statistics, '`.stt3`':
  `id true body state x y z ver_qty edge_qty face_qty vol xc yc zc`

`xc`, *`yc`* and *`zc`* denote the centroid (centre of mass) coordinates.

2.2.4 Post-Processing

- polyhedron identifier file: '`.polyid`'

  It contains the identifiers of the polyhedra of which specific points belong (see option '`-pointpoly`'). By definition, they range from `1` to the maximum number of polyhedra in the tessellation. In the case of a point which does not belong to any polyhedron, the returned value is `0`.
- polyhedron neighbours file: '`.neigh`'

  It contains information on the neighbours of the polyhedra. The file is formatted with one polyhedron per line, with the following entries: *`id neighbour_qty neighbour1_id neighbour2_id ... neighbour1_facearea neighbour2_facearea ... neighbour1_faceeq neighbour2_faceeq ...`*. *`neighbour#_id`* is a positive integer, except for a polyhedron face which belongs to the boundary of the domain (in this case, the value ranges between -6 and -1). A face equation is specified by the parameters d, a, b and c (in this order), with the equation being: $ax + by + cz = d$. The vector (a, b, c) is pointing outwards of the polyhedron.

2.3 Examples

Below are some examples of use of neper -T. Illustrations can be found at `http://neper.sourceforge.net/neper_t.html`.

1. Generate a polycrystal made of 100 grains, with identifier = 1.

   `$ neper -T -n 100 -id 1`
2. Use an elongated domain and generate a polycrystal made of 100 grains, with identifier = 1.

   `$ neper -T -n 100 -id 1 -domain cube 3 1 0.33`
3. Generate a polycrystal made of 100 grains, with identifier = 1, and stretch it to model a morphological texture.

   `$ neper -T -n 100 -id 1 -scale 3 1 0.33`
4. Generate a polycrystal made of 100 columnar grains.

   `$ neper -T -n 100 -id 1 -morpho columnarz`
5. Generate a polycrystal made of approximately 5x5x5 truncated octahedra.

   `$ neper -T -n 5 -morpho tocta`
6. Generate a polycrystal made of 100 grains in a cylindrical domain of height 2 and diameter 1.

   `$ neper -T -n 100 -id 1 -domain cylinder 2 1`

3 Tessellation Free Meshing: neper -FM

Module -FM is the module to generate a free mesh of a tessellation, that is, a mesh comprised of tetrahedral elements that conform to the tessellation morphology. The aim is to generate a mesh into elements of size as close as possible to a desired target value, and of high quality, that is, of equilateral shape. The input file is a tessellation file ('`.tess`'), as provided by module -T. The output mesh can be written in several formats.

Several options are available for specifying the desired mesh properties. The target element size of the mesh can be specified through the following parameters:

- The *characteristic length* (`cl`). It corresponds to the target size of the elements. This size is the length of a line element (1D), and the length of the edge of a triangle element (2D) and of a tetrahedral element (3D). For convenience, a value relative to the average polyhedron size, `rcl`, is also defined: $\texttt{rcl} = 2 * \texttt{cl}/(\texttt{average\_poly\_volume}^{1/3})$.

For ensuring mesh quality to the greatest extent possible, Neper includes several advanced capabilities:

- Geometry regularization. It consists in removing the small features of the tessellation (the edges and faces), which are smaller than the target element size and as a consequence would need local mesh over-refinements. Using this capability is done by allowing some level of geometrical distortion, the face *flatness fault*, through option '`-maxff`' (value in degree).
- Multimeshing. Each tessellation face and volume is meshed separately of the others, with several meshing algorithms, and to the mesh of best quality is retained. This is needed for meshing Voronoi tessellations, and has the advantage of ensuring meshing robustness and optimizing mesh quality. This is controlled by options '`-mesh2dalgo`' and '`-mesh3dalgo`'.
- Remeshing can also be applied to generate a new, good-quality mesh on a mesh containing poor-quality elements (options starting by '`-remesh`'). The variables defined on the old mesh can be transported on the new mesh (options starting by '`-transport`').

Mesh partitioning capabilities enable to divide the mesh nodes and elements into several sets while minimizing the interfaces between them[1], for parallel finite element calculations. Partitioning can return any number of partitions, or more efficiently, can be carried out according to a given parallel computer architecture, in which case the number of partition must be a power of 2 (options starting by '`-part`').

In the output mesh, the individual entities of the tessellations (the volumes, the faces, the edges and the vertices) are described by element sets (option '`-outdim`'). Node sets of the faces, the edges and the vertices of the surface of the tessellation are also provided for prescribing the boundary conditions (option '`-nset`'). The surface element sets (triangles) are also provided (option '`-faset`'). The mesh order can be 1 or 2, corresponding to 3-node tetrahedral elements and 10-node tetrahedral elements, respectively (option '`-order`'). Statistical data can be obtained on the properties of the tessellations and meshes (options starting by '`-stat`').

Options are also available to work on an existing mesh (options starting by '`-loadmesh`').

[1] Each partition being assigned to a processor in the finite element calculation, the minimization of the interfaces between the partitions is done in terms of the number of necessary communications between processors.

Here is what a typical run of module -FM looks like:

```
$ neper -FM n10-id1.tess -maxff 20

========================    N   e   p   e   r    ======================
Info   : A 3D random polycrystal generator for the finite element method
Info   : Version 1.10.0
Info   : (compiled with: gsl libmatheval)
Info   : Loading initialization file '/home/rquey/.neperrc'...
Info   : ---------------------------------------------------------------
Info   : MODULE -FM loaded with arguments:
Info   : [ini file] -nset faces -mo 2 -gmsh /home/rquey/bin/gmsh -surf z1
         -partrenumber 1
Info   : [com line] n10-id1.tess -maxff 20
Info   : ---------------------------------------------------------------
Info   : Reading input data ...
Info   :   - Reading arguments ...
Info   : Creating geometry ...
Info   :   - Loading tessellation ...
Info   :     [i] Parsing file 'n10-id1.tess' ...
Info   :     [i] Parsed file 'n10-id1.tess'.
Info   :   - Regularizing tessellation ... (sel = 0.12)
Info   :     > loop    length   deleted
Info   :     >    1      100%        16
Info   :     >    2      100%         0
Info   : Meshing ... (cl = 0.232, pl = 2)
Info   :   - Preparing  ...  100%
Info   :   - 0D meshing ...  100%
Info   :   - 1D meshing ...  100%
Info   :   - 2D meshing ...  100% (0.26|0.85/16%|83%)
Info   :   - 3D meshing ...  100% (0.89|0.91/100%| 0%)
Info   :   - Switching mesh to order 2 ...
Info   :   - Searching nsets ...
Info   : Writing mesh results ...
Info   :   - Preparing mesh ...
Info   :   - Mesh properties:
Info   :     > Node number:     1852
Info   :     > Elt  number:     1041
Info   :     > Mesh volume:    1.000
Info   :   - Writing mesh ...
Info   :     [o] Writing file 'n10-id1.msh' ...
Info   :     [o] Wrote file 'n10-id1.msh'.
Info   : Elapsed time: 13.353 secs.
========================================================================
```

3.1 Arguments

3.1.1 Input Data

In normal use, the input data is a tessellation file:

`file_name` [Input data]
Name of the tessellation file.
Possible values: `any`. Default value: `none`.

It is also possible to load a mesh from a file. (Using option '`-o`' along with this capability avoids overwriting the input data.)

`-loadmesh file_name` [Input data]
Load a mesh from a file ('`.msh`' format).
Possible values: `any`. Default value: `none`.

`-loadmeshnodecoo file_name` [Input data]
Overwrite the node coordinates. The file must contain the list of coordinates (3 real values per node).
Possible values: `any`. Default value: `none`.

3.1.2 General Options

`-gmsh full_path_name` [Requirement]
Specify the *full* path of the Gmsh binary.
Possible values: `any`. Default value: `/usr/local/bin/gmsh`.

`-o file_name` [Option]
Specify output file name.
Possible values: `any`. Default value: `none`.

3.1.3 Geometry Regularization Options

A non-zero value of *maxff* is necessary to enable geometry regularization; the other options are for fine tuning.

`-maxff real` [Option]
Maximum face flatness fault which is allowed (in degree).
Possible values: `0 to 180 (recommended: 20)`. Default value: `0`.

`-sel or -rsel real` [Secondary option]
Absolute or Relative Small Edge (maximum) Length. The relative small edge length is defined relative to the default value. By default, *sel* is set so as to avoid mesh over-refinement (*cl*/*pl*). Use this option if you want to choose a different length.
Possible values: `any`. Default value: `-sel` *cl*/*pl*.

`-mloop integer` [Secondary option]
Maximum number of edge deletion loops.
During each loop, the small edges are considered in turn from the shortest to the largest. One loop already leads to very satisfactory results. Use more to get better results. The deletion process completes as soon as no edges are deleted within a loop.
Possible values: `any`. Default value: `2`.

3.1.4 Meshing and Multimeshing Options

`-cl or -rcl` ***real*** [Option]

Absolute or relative characteristic length of the elements. `rcl` is defined relative to the average polyhedron volume. The default `-rcl 1` leads to a mesh density of about 100 tetrahedral elements per grain.
Possible values: `any`. Default value: `-rcl 1`.

`-dim` ***integer*** [Option]

Specify the mesh dimension.
Possible values: `0 to 3`. Default value: `3`.

`-order` ***integer*** [Option]

Specify the mesh order.
Possible values: `1 or 2`. Default value: `1`.

`-pl` ***real*** [Secondary option]

Progression factor for the element characteristic lengths. This value is the maximum ratio between the lengths of two adjacent 1D elements.
Possible values: `any >= 1`. Default value: `2`.

`-cl3 or -rcl3` ***real real real*** [Secondary option]

Absolute or relative characteristic length of the elements in the x, y and z directions. `rcl3` is defined relative to the average polyhedron volume. Note that options '`-[r]cl`' and '`-[r]cl3`' are mutually exclusive.
Possible values: `any`. Default value: `none`.

`-clmin` ***real*** [Secondary option]

Minimum characteristic length of the elements. Using this option is not recommended.
Possible values: `any`. Default value: `none`.

The following options define the 2D and 3D-meshing algorithms. The algorithms have the format '*mesh*' or '*mesh/opti*', where *mesh* and *opti* stand for the meshing and optimization algorithms and are 4-character long. *Multimeshing* can be used by providing several algorithms combined with commas, e.g. *mesh1/opti1,mesh1/opti2,mesh2/opti2*. The 2D and 3D meshings are carried out using the Gmsh[1] and Netgen[2] libraries (see the Gmsh reference manual for information on the algorithms).

For 2D meshing, the available values of *mesh* are `made` (MeshAdapt+Delaunay), `mead` (MeshAdapt), `dela` (Delaunay) and `fron` (Frontal). There is no optimization. The default is `fron,made` and it is recommended to retain multimeshing in use for meshing robustness sake. For 3D meshing, the available values of *mesh* are currently limited to `netg` (Netgen). The available values of *opti* are `gmsh` (Gmsh), `netg` (Netgen) and `gmne` (Gmsh+Netgen). For convenience, two generic entries are also defined. The entry `default`, which is the default value, provides a good balance between mesh quality and computation time. The entry `qualmax` provides the best results on mesh quality (full use of multimeshing). The values of `default` are `fron,made` for the 2D case and `netg/gmsh,netg/gmne` for the 3D case. The values of `qualmax` are `made,mead,dela,fron` for the 2D case and `netg/gmsh,netg/netg,netg/gmne` for the 3D case.

[1] Ch. Geuzaine and J.-F. Remacle, Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities, International Journal for Numerical Methods in Engineering, 79, 1309–1331, 2009.

[2] J. Schoberl, Netgen, an advancing front 2d/3d-mesh generator based on abstract rules. Comput. Visual. Sci., 52, 1–41, 1997.

-mesh2dalgo *char_string* [Option]

Specify the 2D meshing algorithm. Multimeshing is allowed by providing several algorithms, (combine with commas).
Possible values: `made`, `mead`, `dela`, `fron`,. Default value: `default (= fron,made)`.

-mesh3dalgo *char_string* [Option]

Specify the 3D meshing algorithm. Multimeshing is allowed by providing several algorithms, (combine with commas).
Possible values: `netg`, `netg/netg`, `netg/gmsh`, `netg/gmne`,. Default value: `default (= netg/gmsh,netg/gmne)`.

-mesh3doptiexpr *string* [Secondary option]

Specify the value of *O* for the multimeshing optimization, as a function of *Odis* and *Osize* (see the Neper paper).
Possible values: `any`. Default value: `Odis^0.8*Osize^0.2`.

-mesh3doptidisexpr *string* [Secondary option]

Specify the value of *Odis* for the multimeshing optimization, as a function of the element distortion parameter *dis* (see the Neper paper).
Possible values: `any`. Default value: `dis^(exp((dis^0.1)/(dis^0.1-1)))`.

3.1.5 Domain Boundary Meshing Options

These options are for specifying geometry regularization and meshing conditions for the polyhedra at the domain boundary different than those that apply to the inner polyhedra. This is useful for coarsening the meshes of the boundary grains when they are disregarded in the analysis due to possible boundary effects. The domain boundary polyhedra can be defined using the following variables:

- *body*: the minimum number of polyhedra between the considered polyhedron and the domain boundary. Its value is 0 for polyhedra intersecting the domain boundary, and increases with the distance to the domain boundary.
- *true*: a polyhedron is said to be *true* if its shape is not biased by the domain boundary. This requires the polyhedron not to be cut by the domain boundary, but actually the condition is a bit more restrictive. The value of *true* is an integer equal to 0 for polyhedra which do not match the above-mentionned criterion, and higher values otherwise. For the latter cases, the value of *true* is defined to be equal to n for true polyhedra surrounded by polyhedra whose value of *true* is higher than or equal to $(n-1)$. This definition is consistent with the one of the *body* variable, in terms of how the values of neighbouring grains compare. As for *body*, the value of *true* increases with the distance to the domain boundary, but typically $true \leq body$.

-dbound *char_string* [Option]

Define which polyhedra belong to the domain boundary. The expression can be based on the following arguments: `body` and `true`. An example is `"body<=1"`.
Possible values: `any`. Default value: `none`.

-dboundcl and -dboundrcl *real* [Option]

Absolute or relative characteristic length of the elements at the domain boundary. `rcl` is defined relative to the average polyhedron volume.
Possible values: `any`. Default value: `none`.

-dboundsel and -dboundrsel *real* [Secondary option]

Absolute or Relative Small Edge (maximum) Length at the domain boundary. The relative small edge length is defined relative to the default value. By default, *sel* is set so as to avoid

mesh over-refinement (*cl*/*pl*). Use this option if you want to choose a different length.
Possible values: `any`. Default value: `-sel` *cl*/*pl*.

`-dboundpl` *real* [Secondary option]
Progression factor for the element characteristic lengths. This value is the maximum ratio between the lengths of two adjacent 1D elements.
Possible values: `any >= 1`. Default value: `2`.

3.1.6 Mesh Partitioning Options

Mesh partitionning is achieved through the libScotch library[3]. In Neper, The two following options enable to turn on mesh partitioning; they are mutually exclusive,

`-partqty` *integer* [Option]
Use this option to specify the quantity of partitions.
Possible values: `any`. Default value: `0`.
Result file: extension '`.[e,n]part`'.

`-partarchfile` *file_name* [Option]
Use this option to specify the architecture of the target machine. Give as argument the name of the file describing the architecture.
Possible values: `any`. Default value: `none`.
Result file: extension '`.[e,n]part`'.

Here are additional options,

`-partbalancing` *real* [Secondary option]
Use this option to set the level of partition balancing (0: none, 1:full). This is a highly CPU-sensitive capability (full balancing requires a lot of time).
Possible values: `0 to 1`. Default value: `0.5`.

`-partmethod` *char_string* [Secondary option]
Specify the partitioning method, expressed in Scotch's jargon.
Possible values: `any (including none)`. Default value: `see_the_source`.

`-partrenumbering` *logical* [Secondary option]
Use this option to renumber the nodes and elements according to partitioning.
Possible values: `0 or 1`. Default value: `0`.

`-partsets` *logical* [Secondary option]
Use this option to print the partitions as nsets and elsets in the mesh file (`geof` format only).
Possible values: `0 or 1`. Default value: `1`.

3.1.7 Remeshing Options

`-remesh` *file_name* [Option]
Use this option for remeshing a mesh. Provide as argument the mesh file.
Possible values: `any`. Default value: `none`.

`-remeshtess` *file_name* [Option]
Use this option to specify a tessellation associated to the mesh to remesh. This can be useful, for example, when the meshed domain is not a regular box, to determine the node sets. Provide as argument the tessellation file.
Possible values: `any`. Default value: `none`.

[3] F. Pellegrini, Scotch and libScotch 5.1 User's Guide, INRIA Bordeaux Sud-Ouest, ENSEIRB & LaBRI, UMR CNRS 5800, 2008.

-transport *file_name integer char_string char_string file_name* ... [Option]
Use this option for transporting data from a parent mesh to a child mesh (typically obtained by remeshing). First provide the name of the parent mesh file. The child mesh is taken as the result mesh (usually obtained by remeshing, but it can also be loaded with '-loadmesh'). Then provide as argument the number of different data to transport; then, for each of them, elt (mandatory), the type of data (under the format [integer,real]X, where X is the dimension) and the name of the file containing the parent data.
Possible values: any. Default value: 0.

-transporttess *file_name* [Option]
Use this option to specify a tessellation associated to the mesh from which the data are transported. This is not mandatory. Provide as argument the tessellation file.
Possible values: any. Default value: none.

3.1.8 Output Options

-outdim *char_string* [Option]
Specify the dimensions of the mesh to output. It can go from 0 to 3, for point to volume elements (combine with commas).
Possible values: 0, 1, 2, 3,. Default value: 0,1,2,3.

-format *char_string* [Option]
Specify the format of the output file(s). For the mesh, the available formats are: msh for Gmsh, abq for Abaqus, geof for Zset/Zébulon, and fev for Fem-Evps. For the tessellation geometry, the available formats are: tess and geo. Combine arguments with commas.
Possible values: anyone of the above list. Default value: msh.

-nset *char_string* [Option]
Specify the node sets to provide, among: faces, edges, vertices for all domain faces, edges and vertices, and facebodies and edgebodies for all face and edge bodies. To get the nset corresponding to individual entities, provide their names, that is, for a cubic domain, [x-z][0,1] for the domain faces, [x-z][0,1][x-z][0,1] for the edges, and [x-z][0,1][x-z][0,1][x-z][0,1] for the vertices. Append 'body' to the name to get only the body nodes of the sets. Combine them with commas.
Possible values: any. Default value: faces.

-faset *char_string* [Option]
Specify the domain surfaces to provide. Use 'faces' for all faces. Combine them with commas. Possible values: faces, [x-z][0,1] (for a cubic domain). Default value: none.

3.1.9 Post-Processing Options

-stattess *logical* [Post-processing]
Provide statistics on the tessellation.
Possible values: 0 or 1. Default value: 0.
Result file: extension '.stt#'.

-statmesh *logical* [Post-processing]
Provide information and statistics on the elements and element sets.
Possible values: 0 or 1. Default value: 0.
Result file: extension '.stm#'.

3.1.10 Advanced Options

These advanced options set running conditions for the mesher.

`-mesh2dmaxtime` *real* [Secondary option]
Maximum processing time allowed to the mesher for meshing a tessellation face (in seconds).
Possible values: `any`. Default value: `1000`.

`-mesh2drmaxtime` *real* [Secondary option]
This option is similar to '`-mesh2dmaxtime`', but the actual maximum time is the product of the maximum processing time of the previous meshings by the value provided in argument.
Possible values: `any`. Default value: `100`.

`-mesh2diter` *integer* [Secondary option]
Maximum iterations in 3D meshing for a particular face (in case of failure).
Possible values: `any`. Default value: `3`.

`-mesh3dmaxtime` *real* [Secondary option]
Maximum processing time allowed to the mesher for meshing a tessellation volume (in seconds).
Possible values: `any`. Default value: `1000`.

`-mesh3drmaxtime` *real* [Secondary option]
This option is similar to '`-mesh3dmaxtime`', but the actual maximum time is the product of the maximum processing time of the previous meshings by the value provided in argument.
Possible values: `any`. Default value: `100`.

`-mesh3diter` *integer* [Secondary option]
Maximum iterations in 3D meshing for a particular volume (in case of failure).
Possible values: `any`. Default value: `3`.

`-mesh3dclconv` *real* [Secondary option]
Maximum tolerated difference between the characteristic length `cl` and the average element length (for each polyhedron). Neper tries its best to get the average element size to match `cl`. Use this option to change the tolerance on the relative difference between the two. This is a highly CPU-sensitive capability (using a high value can be an efficient way to speed up meshing).
Possible values: `any`. Default value: `0.02`.

3.2 Output Files

3.2.1 Mesh

The mesh can be written in the following formats:

- Gmsh format: file '`.msh`'
- Abaqus format: file '`.inp`'
- Zset/Zébulon format: file '`.geof`'
- Fem-Evps format: files '`.parms`', '`.mesh`', '`.surf`', '`.opt`' and '`.bcs`'

The following files are for describing the partitions:

- Node partition description, file '`.npart`': *node_id partition_id*. The partition identifier ranges from 1 to the total number of partitions.
- Element partition description, file '`.epart`': *elt_id partition_id*. The partition identifier ranges from 1 to the total number of partitions.
- Remeshing file, '`.rem`': *elt_id corresponding_old_elt_id*.

3.2.2 Tessellation

- Neper-native tessellation file: '`.tess`'

  It contains an exhaustive description of the tessellation. See Appendix A [File Formats], page 43 for the file syntax.
- Gmsh geometry file: '`.geo`'

  It contains a minimal description of the tessellation, written under the Gmsh geometry file format '`.geo`'. This file can be opened with Gmsh for visualization. If the tessellation has been regularized, Gmsh will complain about surfaces not being plane, but for visualization this can be disregarded. (Note that you can even get a mesh of the tessellation out from Gmsh, but it will be of lower quality than by using module `-FM`.)
- Ply file: '`.ply`'

  It contains a description of the tessellation written under the standard "Polygon File Format" '`.ply`'.

3.2.3 Statistics

Several files are provided for statistics on tessellations, whose formats are provided below. All files are formatted with one entity (vertex, edge, face or polyhedron) per line.

- Tessellation vertex statistics, '`.stt0`': *`id true body state x y z`*
- Tessellation edge statistics, '`.stt1`': *`id true body state length`*
- Tessellation face statistics, '`.stt2`': *`id true body state ver_qty area ff`*
- Tessellation polyhedron statistics, '`.stt3`':
  `id true body state x y z ver_qty edge_qty face_qty vol xc yc zc`

`xc`, *`yc`* and *`zc`* denote the centroid (centre of mass) coordinates.

The following are for statistics on the mesh.

- Mesh non-quality statistics file, '`.stm1`': *`id elset_id true vol mean_length x y z`*.
- Mesh quality statistics file, '`.stm2`': *`id elset_id true radius_ratio angle_min`*.

3.3 Examples

Below are some examples of use of neper -FM. Illustrations can be found at `http://neper.sourceforge.net/neper_fm.html`.

1. Mesh tessellation n100-id1.tess.

   ```
   $ neper -FM n100-id1.tess
   ```
2. Mesh tessellation n100-id1.tess using geometry regularization.

   ```
   $ neper -FM n100-id1.tess -maxff 20 -sel 0.05
   ```
3. Mesh tessellation n100-id1.tess with a mesh size of rcl = 0.5 and in 2nd-order elements.

   ```
   $ neper -FM n100-id1.tess -maxff 20 -sel 0.05 -rcl 0.5 -order 2
   ```
4. Mesh tessellation n100-id1.tess with small elements for the inner grains and bigger elements for the surface grains.

   ```
   $ neper -FM n100-id1.tess -maxff 20 -rcl 0.2 -dbound "body==0"
     -dboundrcl 0.5 -dboundsel 0.05
   ```
5. Remesh mesh n150_def.msh (comprising poor-quality elements) into a clean, new mesh. Transport the scalar data of file 'n150_def.data' from the deformed mesh to the new mesh.

   ```
   $ neper -FM -remeshtess n150.tess -remesh n150_def.msh -transport
     n150_def.msh 1 elt real1 n150_def.data -rcl 0.5 -o n150_new
   ```

   Below are some examples of advanced use.

1. *Multimeshing.* Improve mesh quality as far as possible through 3D multimeshing.

   ```
   $ neper -FM n10-id1.tess -maxff 20 -rcl 0.8 -order 2
     -format tess,msh,abq -o n10
     -mesh3dalgo netg/gmsh,netg/netg,netg/gmne
   ```

2. *Mesh partitionning.* Like 2., but partition in 8 partitions and renumber the nodes and elements accordingly.

   ```
   $ neper -FM n10-id1.tess -maxff 20 -rcl 0.8 -order 2
     -format tess,msh,abq -o n10
     -partq 8 -partrenum 1
   ```

3. *Orthotropic mesh refinement.* Mesh tessellation '`n10-id1.tess`' with different element characteristic lengths along x, y and z: *rcl* = 1, *rcl* = 0.5 and *rcl* = 0.25, respectively. Use geometry regularization.

   ```
   $ neper -FM n10-id1.tess -rcl3 1 0.5 0.25 -maxff 20
   ```

4. *Data transport 1/2.* Like 7., and transport data defined on '`n10.msh`' to '`n10-b.msh`'. The data are 3D vectors given in file '`data`' (format = 3 coordinates per line).

   ```
   $ neper -FM -remesh n10.msh -remeshtess n10.tess -rcl 0.5
     -transport n10.msh 1 elt real3 data -o n10-b
   ```

5. *Data transport 2/2.* Transport data from an existing old mesh '`n10.msh`' to an existing new mesh '`n10-b.msh`'. The data are 3D vectors given in file '`data`' (format = 3 coordinates per line).

   ```
   $ neper -FM -loadmesh n10-b.msh
     -transport n10.msh 1 elt real3 data -o n10-b
   ```

4 Tessellation Mapped Meshing: neper -MM

Module -MM is the module to generate a mapped mesh of a tessellation, that is, a mesh comprised of regular, brick elements. Such a mesh does not conform exactly to the tessellation morphology: the interfacial features, and more particularly the grain boundaries and triple lines, have stepped shapes. The input file is a tessellation file ('`.tess`'), as provided by module -T (is which case the domain must be cuboidal), or simply the data `(n, id)` (same input as for module -T, see Chapter 2 [Module -T], page 7). The output mesh can be written in several formats.

In addition to the tessellations generated by module -T (or equivalently through the `(n, id)` data), two other types of tessellation can be obtained: *periodic tessellations*, whose grains show periodicity conditions at the domain boundary, and *subdomain-type tessellations*. The latter are cut out from tessellations of larger domains, and which have the same polyhedron volume density. Thus, the tessellations contain grains whose centres are not within the domain. This behaviour is controlled by option '`-ttype`'.

The level of mesh density is specified a bit differently than in module -FM. This is done through the following parameter:

- The number of elements along one dimension of the domain (`msize`). For a non-cubic domain, an average, equivalent length of the domain is considered so as to get cubic elements. The number of elements along each of the directions of the domain can also be specified explicitly (option '`-msize3`').

In the output mesh, the grains are described by element sets. Node sets of the faces, the edges and the vertices of the surface of the tessellation are also provided for prescribing the boundary conditions (option '`-nset`'). The surface element sets (squares) are also provided (option '`-faset`'). The mesh order can be 1 or 2, corresponding to 8-node cubic elements and 20-node cubic elements, respectively (option '`-order`').

Here is what a typical run of module -MM looks like:

```
$ neper -MM n10-id1.tess

========================    N   e   p   e   r    =======================
Info   : A 3D random polycrystal generator for the finite element method
Info   : Version 1.10.0
Info   : (compiled with: gsl libmatheval)
Info   : Loading initialization file '/home/rquey/.neperrc'...
Info   : ---------------------------------------------------------------
Info   : MODULE -MM loaded with arguments:
Info   : [ini file] (none)
Info   : [com line] n10-id1.tess
Info   : ---------------------------------------------------------------
Info   : Mapped meshing ...
Info   :   - Loading tessellation ...
Info   :     [i] Parsing file 'n10-id1.tess' ...
Info   :     [i] Parsed file 'n10-id1.tess'.
Info   :   - Generating mesh ...
Info   :   - Searching elsets ...  100%
Info   : Writing results ...
Info   :   - Writing mesh results ...
Info   :     [o] Writing file 'n10-id1.msh' ...
Info   :     [o] Wrote file 'n10-id1.msh'.
```

```
Info   : Elapsed time: 0.033 secs.
========================================================================
```

4.1 Arguments

4.1.1 Input Data

The required input data are:

file_name [Input data]
Name of the tessellation file.
Possible values: `any`. Default value: `none`.

or, the following ones,

`-n` *integer* [Input data]
Number of polyhedra of the tessellation, except for regular morphologies (truncated octahedra, etc., see option '`-morpho`' for details).
Possible values: `any`. Default value: `none`.

`-id` *integer* [Input data]
Identifier of the tessellation.
Possible values: `any`. Default value: *random*.

`-morpho` *char_string* [Input data]
Type of morphology of the polyhedra. For random Voronoi tessellations, it can be either equiaxed (`equiaxed`), columnar (`columnar{x,y,z}`) or bamboo-like (`bamboo{x,y,z}`).
To load a particular set of polyhedron centres, use the syntax '`@`*file_name*' where *file_name* is the name of a file containing the $3 * n$ coordinates.
Regular morphologies can also be obtained: cubes (`cube`) or truncated octahedra (`tocta`), in which case the value of option '`-n`' stands for the number of polyhedra along an edge of the domain instead of the total number of polyhedra.
Possible values: `see above list`. Default value: `equiaxed`.

4.1.2 General Options

`-o` *file_name* [Option]
Specify output file name.
Possible values: `any`. Default value: `none`.

4.1.3 Domain Options

`-domain` *char_string ...* [Option]
Specify the type of domain and its size. The only type of domain available is of cuboidal shape ('`cube`'). Then provide the sizes along the x, y and z directions.
Possible values: `see above list`. Default value: `cube 1 1 1`.

4.1.4 Tessellation Options

This is for a tessellation mesh built from (`n`, `id`) only, not from a tessellation file.

`-scale` *real real real* [Option]
Specify the factors in the x, y and z directions by which the tessellation is to be scaled (once generated).
Possible values: `any`. Default value: `none`.

`-ttype` *char_string* [Option]
Specify the type of tessellation (applies to tessellations built with `(n, id)`, not by module -T (‘`.tess`’ file)). The available values are ‘`standard`’ for a standard tessellation, `periodic` for a periodic tessellation and `subdomain` for a subdomain-type tessellation.
Possible values: `see above list`. Default value: `standard`.

4.1.5 Mesh Options

`-msize` *integer* [Option]
Specify the mesh size (number of elements per unit length).
Possible values: `any`. Default value: `20`.

`-msize3` *integer integer integer* [Secondary option]
Specify the mesh size (number of elements per unit length) along the x, y and z directions.
Possible values: `any`. Default value: `20 20 20`.

`-order` *integer* [Option]
Specify the mesh order.
Possible values: `1 to 2`. Default value: `1`.

4.1.6 Output Options

`-outdim` *char_string* [Option]
Specify the dimensions of the mesh to output. It can go from 0 to 3, for point to volume elements (combine with commas).
Possible values: `0, 1, 2, 3,`. Default value: `3`.

`-nset` *char_string* [Option]
Specify the node sets to provide, among: `faces`, `edges`, `vertices` for all domain faces, edges and vertices, and `facebodies` and `edgebodies` for all face and edge bodies. To get the nset corresponding to individual entities, provide their names, that is, for a cubic domain, `[x-z][0,1]` for the domain faces, `[x-z][0,1][x-z][0,1]` for the edges, and `[x-z][0,1][x-z][0,1][x-z][0,1]` for the vertices. Append ‘`body`’ to the name to get only the body nodes of the sets. Combine them with commas.
Possible values: `any`. Default value: `faces`.

`-faset` *char_string* [Option]
Specify the domain surfaces to provide. Use ‘`faces`’ for all faces. Combine them with commas.
Possible values: `faces, [x-z][0,1]`. Default value: `none`.

4.2 Output Files

4.2.1 Mesh

The mesh can be written in the following formats:

- Gmsh format: ‘`.msh`’
- Abaqus format: file ‘`.inp`’
- Zset/Zébulon format: ‘`.geof`’

When the input data is of type `(n, id)`, the following file is also generated (as in module -T):

- orientation input file: ‘`.oin`’

  It contains data for generating the grain orientations, and is an input file for module -O (see Chapter 5 [Module -O], page 27).

4.3 Examples

Below are some examples of use of neper -MM. Illustrations can be found at http://neper.sourceforge.net/neper_mm.html.

1. Mesh tessellation n100-id1.tess.

   ```
   $ neper -MM n100-id1.tess
   ```

2. Mesh tessellation n100-id1.tess with 40 elements per unit length.

   ```
   $ neper -MM n100-id1.tess -msize 40
   ```

3. Provide a mesh of a periodic tessellation made of 100 grains, with identifier 1.

   ```
   $ neper -MM -n 100 -id 1 -msize 40 -ttype periodic
   ```

5 Crystallographic Orientation Generation: neper -O

Module -O is the module to generate crystallographic orientations for the grains of the tessellations generated by module -T. The orientations are randomly distributed according to a uniform distribution. They can be provided according to different descriptors: Euler angles (Bonge, Kocks and Roe conventions), rotation matrix, rotation axis / angle, Rodrigues vector and quaternion. The input data is a file '`.oin`' provided by module -T (or module -MM), but it can also be the data (`n`, `id`). The output data in an orientation file '`.ori`'. Module -O also provides capabilities to generate colours from the orientations (useful for module -VS).

Here is what a typical run of module -O looks like:

```
$ neper -O -n 10 -id 1

========================    N   e   p   e   r    ======================
Info   : A 3D random polycrystal generator for the finite element method
Info   : Version 1.10.0
Info   : (compiled with: gsl libmatheval)
Info   : Loading initialization file '/home/rquey/.neperrc'...
Info   : ---------------------------------------------------------------
Info   : MODULE  -O loaded with arguments:
Info   : [ini file] -crysym cubic
Info   : [com line] -n 10 -id 1
Info   : ---------------------------------------------------------------
Info   :     [o] Writing file 'n10-id1.ori' ...
Info   :     [o] Wrote file 'n10-id1.ori'.
Info   : Elapsed time: 0.001 secs.
=======================================================================
```

5.1 Arguments

5.1.1 Input Data

The required input data are:

file.oin [Input data]
Name of the input file.
Possible values: `any`. Default value: `none`.

or, the two following ones:

`-n` ***integer*** [Input data]
Number of crystallographic orientations.
Possible values: `any`. Default value: `none`.

`-id` ***integer*** [Input data]
Identifier of the set of orientations.
Possible values: `any`. Default value: `none`.

Alternatively, orientations can be loaded from a file,

`-load` ***input_type file_name*** [Input data]
Load an orientation file. Provide the type of orientation descriptor (see option '`-descriptor`') and the file name.
Possible values: `any any`. Default value: `none`.

5.1.2 General Options

`-o` *`file_name`* [Option]

Specify orientation output file name.
Possible values: `any`. Default value: `none`.

5.1.3 Orientation Options

`-crysym` *`char_string`* [Secondary option]

Specify the crystal symmetry. This is only used to reduce the domain of definition of the orientation descriptors.
Possible values: `triclinic or cubic`. Default value: `triclinic`.

5.1.4 Output Options

`-descriptor` *`char_string`* [Option]

Select the orientation descriptor. It can be Euler angles in Bunge, Kocks or Roe convention (`e`, `ek`, `er`), rotation matrix (`g`), axis / angle or rotation (`rtheta`), Rodrigues vector (`R`) or quaternion (`q`).
Possible values: `above-mentionned values`. Default value: `e`.

`-format` *`character_string`* [Option]

Specify the format of output file(s). The available formats are: the Neper-native `plain` (i.e. only the descriptors on successive lines), the Zset/Zébulon `geof` and the Fem-Evps `fev` (combine with commas).
Possible values: `above-mentionned values`. Default value: `plain`.

5.1.5 Colouring Options

`-colour` *`character_string`* [Option]

Use this option to get colours from the orientations. Provide as argument the type of colouring: the only one available is from the Rodrigues vectors (`R`). To use this option, '`-crysym`' must be set to cubic.
Possible values: `R`. Default value: `none`.
Result file: extension '`.col`'.

5.2 Output Files

- Crystal orientation file, '`.ori`': format corresponding to option '`-format`'. The grains orientations are listed on successive lines.
- Orientation colour file, '`.col`': *`red_level green_level blue_level`*. The levels are integers comprised in the range [0, 255].

5.3 Examples

Below are some examples of use of neper -O. Illustrations can be found at `http://neper.sourceforge.net/neper_o.html`.

1. Generate a set of 100 "random" crystallographic orientations, with identifier = 1. Note the uniform distribution.

```
$ neper -O -n 100 -id 1
```

Below are additional examples.

1. *Orientation generation.* Generate a set of crystallographic orientations from the input file '`n100-id1.oin`'.

```
$ neper -O n100-id1.oin
```

2. *Orientation colour generation.* Generate colours which correspond to the orientations written in file '`n100-id1.ori`' (Euler angles in Kocks convention).

   ```
   $ neper -O -load ek n100-id1.ori -crysym cubic -col R
   ```

6 Mesh and Data Visualization: neper -VS

Module -VS is the Neper visualization module, with which the tessellations and meshes can be rendered as publication-quality images. It is also possible to visualize data on them using colours, or displacements of the nodes and to plot data on slices of the mesh. The output file is a PNG image file. Although this module has limited capabilities (compared to visualization applications such as Paraview, etc.), it can be useful for rapid and non-interactive rendering.

Contrary to the other modules, module -VS executes the provided arguments one after the other. Typically, using module -VS consists in loading a tessellation and / or a mesh (options starting by '`-load`'), then data fields to render them. The data can apply to the tessellation entities: polyhedra, faces, edges and vertices, and to the mesh entities: 3D elements, 2D elements, 1D elements, 0D elements and nodes (options starting by '`-data`'). The entities that are to be visible on the rendered image, for example particular tessellation polyhedra, element sets or elements, can also be specified (options starting by '`-show`'). The way they are plotted: camera position and angle, projection type, image size, etc., can be set up as well (options starting by '`-camera`' or '`-image`'). The POV-Ray ray-tracing library is used for generating the images.

Here is what a typical run of module -VS looks like:

```
$ neper -VS -loadtess n10-id1.tess -loadmesh n10-id1.msh \
            -dataelsetcol ori=n10-id1.ori -print img

========================    N   e   p   e   r    ======================
Info   : A 3D random polycrystal generator for the finite element method
Info   : Version 1.10.0
Info   : (compiled with: gsl libmatheval)
Info   : Loading initialization file '/home/rquey/.neperrc'...
Info   : ---------------------------------------------------------------
Info   : MODULE -VS loaded with arguments:
Info   : [ini file] -imageantialias 3
Info   : [com line] -loadtess n10-id1.tess -loadmesh n10-id1.msh
         -dataelsetcol ori=n10-id1.ori -print img
Info   : ---------------------------------------------------------------
Info   : Loading tessellation ...
Info   :     [i] Parsing file 'n10-id1.tess' ...
Info   :     [i] Parsed file 'n10-id1.tess'.
Info   : Loading mesh ...
Info   :     [i] Parsing file 'n10-id1.msh' ...
Info   :     [i] Parsed file 'n10-id1.msh'.
Info   : Reconstructing mesh ...
Info   :   - Reconstructing 2D mesh ...  100%
Info   :   - Reconstructing 1D mesh ...  100%
Info   :   - Reconstructing 0D mesh ...  100%
Info   : Reading data (elset3d, col)...
Info   :     [i] Parsing file 'n10-id1.ori' ...
Info   : Printing image ...
Info   :     [o] Writing file 'img.pov' ...
Info   :   - Printing mesh ...
Info   :     > Preparing mesh data ...
Info   :     > Reducing data ...
Info   :       . Number of 3D elts    reduced by   0% (to 1000).
Info   :       . Number of elt faces  reduced by  90% (to 600).
```

```
Info   :       . Number of face edges reduced by   0% (to 2400).
Info   :      [o] Wrote file ‘img.pov’.
Info   :    - Generating png file (1200x900 pixels)...
Info   :      [o] Writing file ‘img.png’ ...
Info   :      [o] Wrote file ‘img.png’.
Info   : Printing scale ...
Info   : Elapsed time: 0.789 secs.
========================================================================
```

6.1 Arguments

6.1.1 Tessellation and Mesh Loading

`-loadtess` *`file_name`* [Option]

Load a tessellation from a file (‘`.tess`’).
Possible values: `any`. Default value: `none`.

`-loadmesh` *`file_name`* [Option]

Load a mesh from a file (must be a ‘`.msh`’).
Possible values: `any`. Default value: `none`.

6.1.2 Tessellation Data Loading and Rendering

The following options enable to define the properties (colour and size) of the tessellation entities (polyhedra, faces, edges and vertices). This can be done either directly, by specifying the property values (e.g. the RGB channel values for colour) or indirectly, e.g. using scalar values that are rendered in colour using a given *olour scheme*. In this case, a scale image is generated in addition to the tessellation image. The scale properties can be set up (minimum, maximum and tick values). The image as ‘`-scale`*`entity`*’ as suffix.

`-datapolycol` *`char_string`* [Option]

Set the colours of the tessellation polyhedra. The argument can be one of the following: the name of a colour that will be used for all polyhedra (see Appendix C [Colours], page 49), the name of a file containing a list of colours (provided as RGB channel values), or a string indicating how the colours can be obtained. The string has the format ‘*`var`*`=`*`file_name`*’, where ‘*`var`*’ can be ‘`ori`’ for crystal orientations or ‘`scal`’ for scalar values, and ‘*`file_name`*’ is the name of the file containing the data. The colour schemes used to derive the colours from the data can be specified with options ‘`-datapolycolscheme`’.
Possible values: `any`. Default value: `white`.

`-datapolycolscheme` *`char_string`* [Option]

Set the colour scheme used to get colours from the data of the tessellation polyhedra loaded with option ‘`-datapolycol`’. The type of colour scheme depends on the type of data. For crystal orientations, the colour scheme can be: only "R" for Rodrigues vector colouring; for scalar data, the colour scheme can be any list of colours.
Possible values: `"R" for crystal orientations and any list of colours for scalars`. Default value: `"R" for crystal orientations and "blue,cyan,yellow,green" for scalars`.

`-datapolyscalemin` *`real`* [Option]

Set the minimum of the scale relative to the "-datapolycol scal=" data.
Possible values: `any`. Default value: `data minimum`.

`-datapolyscalemax` *real* [Option]
Set the maximum of the scale relative to the "-datapolycol scal=" data.
Possible values: `any`. Default value: `data maximum`.

`-datapolyscaleticks` *real* [Option]
Set the ticks of the scale relative to the "-datapolycol scal=" data. Provide a string composed of values separated by commas. (Use '`_`' to get a blank space.)
Possible values: `any`. Default value: `none`.

`-datafacecol` *char_string* [Option]
Set the colours of the tessellation faces. See option '`-datapolycol`' for the argument format.
Possible values: `any`. Default value: `white`.

`-datafacecolscheme` *char_string* [Option]
Set the colour scheme used to get colours from the data of the tessellation faces loaded with option '`-datafacecol`'. See option '`-datapolycolscheme`' for the argument format.
Possible values: `see option '-datapolycolscheme'`. Default value: `see option '-datapolycolscheme'`.

`-datafacescalemin` *real* [Option]
Set the minimum of the scale relative to the "-datafacecol scal=" data.
Possible values: `any`. Default value: `data minimum`.

`-datafacescalemax` *real* [Option]
Set the maximum of the scale relative to the "-datafacecol scal=" data.
Possible values: `any`. Default value: `data maximum`.

`-datafacescaleticks` *real* [Option]
Set the ticks of the scale relative to the "-datafacecol scal=" data. Provide a string composed of values separated by commas. (Use '`_`' to get a blank space.)
Possible values: `any`. Default value: `none`.

`-dataedgerad` *char_string* [Option]
Set the radii of the tessellation edges. The argument can be one of the following: a real value that will be used for all entities or the name of a file containing a list of radii.
Possible values: `any`. Default value: `proportional to the polyhedron size`.

`-dataedgecol` *char_string* [Option]
Set the colours of the tessellation edges. See option '`-datapolycol`' for the argument format.
Possible values: `any`. Default value: `black`.

`-dataedgecolscheme` *char_string* [Option]
Set the colour scheme used to get colours from the data of the tessellation edges loaded with option '`-dataedgecol`'. See option '`-datapolycolscheme`' for the argument format.
Possible values: `see option '-datapolycolscheme'`. Default value: `see option '-datapolycolscheme'`.

`-dataedgescalemin` *real* [Option]
Set the minimum of the scale relative to the "-dataedgecol scal=" data.
Possible values: `any`. Default value: `data minimum`.

`-dataedgescalemax` *real* [Option]
Set the maximum of the scale relative to the "-dataedgecol scal=" data.
Possible values: `any`. Default value: `data maximum`.

-dataedgescaleticks *real* [Option]

Set the ticks of the scale relative to the "-dataedgecol scal=" data. Provide a string composed of values separated by commas. (Use '_' to get a blank space.)
Possible values: `any`. Default value: `none`.

-dataverrad *char_string* [Option]

Set the radii of the tessellation vertices. See option '`-dataedgerad`' for the argument format.
Possible values: `any`. Default value: `proportional to the polyhedron size`.

-datavercol *char_string* [Option]

Set the colours of the tessellation vertices. See option '`-datapolycol`' for the argument format.
Possible values: `any`. Default value: `black`.

-datavercolscheme *char_string* [Option]

Set the colour scheme used to get colours from the data of the tessellation vertices loaded with option '`-datavercol`'. See option '`-datapolycolscheme`' for the argument format.
Possible values: `see option '-datapolycolscheme'`. Default value: `see option '-datapolycolscheme'`.

-dataverscalemin *real* [Option]

Set the minimum of the scale relative to the "-datavercol scal=" data.
Possible values: `any`. Default value: `data minimum`.

-dataverscalemax *real* [Option]

Set the maximum of the scale relative to the "-datavercol scal=" data.
Possible values: `any`. Default value: `data maximum`.

-dataverscaleticks *real* [Option]

Set the ticks of the scale relative to the "-datavercol scal=" data. Provide a string composed of values separated by commas. (Use '_' to get a blank space.)
Possible values: `any`. Default value: `none`.

6.1.3 Mesh Data Loading and Rendering

The following options enable to define the properties (colour, size, etc.) of the mesh entities (3D, 2D, 1D and 0D elements, nodes). This can be done either directly, by specifying the property values (e.g. the RGB channel values for colour) or indirectly, e.g. using scalar values that are rendered in colour using a given *colour scheme*. In this case, a scale image is generated in addition to the mesh image. The scale properties can be set up (minimum, maximum and tick values). The image as '`-scale`*`entity`*' as suffix.

The following options enable to load data relative to the 3D mesh elements. Note that `elt3d` can be abbreviated to `elt` in the option names, and that the options can be applied to element sets instead of elements by changing '`elt`' to '`elset`'.

-dataelt3dcol *char_string* [Option]

Set the colours of the 3D elements. The argument can be one of the following: the name of a colour that will be used for all elements (see Appendix C [Colours], page 49), the name of a file containing a list of colours (provided as RGB channel values), or a string indicating how the colours can be obtained. The string has the format '*var=file_name*', where '*var*' can be '`ori`' for crystal orientations or '`scal`' for scalar values, and '*file_name*' is the name of the file containing the data. The colour schemes used to derive the colours from the data can be specified with options '`-dataelt3dcolscheme`'.
Possible values: `any`. Default value: `white`.

`-dataelt3dcolscheme` *char_string* [Option]
Set the colour scheme used to get colours from the data of the 3D elements loaded with option '`-dataelt3dcol`'. The type of colour scheme depends on the type of data. For crystal orientations, the colour scheme can be: only "R" for Rodrigues vector colouring; for scalar data, the colour scheme can be any list of colours.
Possible values: `"R" for crystal orientations and any list of colours for scalars`. Default value: `"R" for crystal orientations and "blue,cyan,yellow,green" for scalars`.

`-dataelt3dscalemin` *real* [Option]
Set the minimum of the scale relative to the "-dataelt3dcol scal=" data.
Possible values: `any`. Default value: `data minimum`.

`-dataelt3dscalemax` *real* [Option]
Set the maximum of the scale relative to the "-dataelt3dcol scal=" data.
Possible values: `any`. Default value: `data maximum`.

`-dataelt3dscaleticks` *real* [Option]
Set the ticks of the scale relative to the "-dataelt3dcol scal=" data. Provide a string composed of values separated by commas. (Use '`_`' to get a blank space.)
Possible values: `any`. Default value: `none`.

`-dataelt3dedgecol` *char_string* [Option]
Set the colours of the edges of the 3D elements. See option '`-dataelt3dcol`' for the argument format.
Possible values: `any`. Default value: `black`.

`-dataelt3dedgecolscheme` *char_string* [Option]
Set the colour scheme used to get colours from the data of the element edges loaded with option '`-dataelt3dedgecol`'. See option '`-dataelt3dcolscheme`' for the argument format.
Possible values: `see option '-dataelt3dcolscheme'`. Default value: `see option '-dataelt3dcolscheme'`.

`-dataelt3dedgerad` *char_string* [Option]
Set the radii of the edges of the 3D elements. The argument can be one of the following: a real value that will be used for all entities, or the name of a file containing a list of radii.
Possible values: `any`. Default value: `proportional to the polyhedron size`.

The following options enable to load data relative to the 2D elements. Note that the options can be applied to element sets instead of elements by changing '`elt`' to '`elset`'.

`-dataelt2dcol` *char_string* [Option]
Set the colours of the 2D elements. See option '`-dataelt3dcol`' for the argument format.
Possible values: `any`. Default value: `black`.

`-dataelt2dcolscheme` *char_string* [Option]
Set the colour scheme used to get colours from the data of the 2D elements loaded with option '`-dataelt2dcol`'. See option '`-dataelt3dcolscheme`' for the argument format.
Possible values: `see option '-dataelt3dcolscheme'`. Default value: `see option '-dataelt3dcolscheme'`.

`-dataelt2dscalemin` *real* [Option]
Set the minimum of the scale relative to the "-dataelt2dcol scal=" data.
Possible values: `any`. Default value: `data minimum`.

-dataelt2dscalemax *real* [Option]
Set the maximum of the scale relative to the "-dataelt2dcol scal=" data.
Possible values: `any`. Default value: `data maximum`.

-dataelt2dscaleticks *real* [Option]
Set the ticks of the scale relative to the "-dataelt2dcol scal=" data. Provide a string composed of values separated by commas. (Use '`_`' to get a blank space.)
Possible values: `any`. Default value: `none`.

-dataelt2dedgecol *char_string* [Option]
Set the colours of the edges of the 3D elements. See option '`-dataelt3dcol`' for the argument format.
Possible values: `any`. Default value: `black`.

-dataelt2dedgecolscheme *char_string* [Option]
Set the colour scheme used to get colours from the data of the edges of the 2D elements loaded with option '`-dataelt2dedgecol`'. See option '`-dataelt3dcolscheme`' for the argument format.
Possible values: `see option '-dataelt3dcolscheme'`. Default value: `see option '-dataelt3dcolscheme'`.

-dataelt2dedgescalemin *real* [Option]
Set the minimum of the scale relative to the "-dataelt2dedgecol scal=" data.
Possible values: `any`. Default value: `data minimum`.

-dataelt2dedgescalemax *real* [Option]
Set the maximum of the scale relative to the "-dataelt2dedgecol scal=" data.
Possible values: `any`. Default value: `data maximum`.

-dataelt2dedgescaleticks *real* [Option]
Set the ticks of the scale relative to the "-dataelt2dedgecol scal=" data. Provide a string composed of values separated by commas. (Use '`_`' to get a blank space.)
Possible values: `any`. Default value: `none`.

-dataelt2dedgerad *char_string* [Option]
Set the radii of the edges of the 3D elements. The argument can be one of the following: a real value that will be used for all entities, or the name of a file containing a list of radii.
Possible values: `any`. Default value: `proportional to the polyhedron size`.

The following options enable to load data relative to the 1D elements. Note that the options can be applied to element sets instead of elements by changing '`elt`' to '`elset`'.

-dataelt1dcol *char_string* [Option]
Set the colours of the 1D elements. See option '`-dataelt3dcol`' for the argument format.
Possible values: `any`. Default value: `black`.

-dataelt1dcolscheme *char_string* [Option]
Set the colour scheme used to get colours from the data of the 1D elements loaded with option '`-dataelt1dcol`'. See option '`-dataelt3dcolscheme`' for the argument format.
Possible values: `see option '-dataelt3dcolscheme'`. Default value: `see option '-dataelt3dcolscheme'`.

-dataelt1dscalemin *real* [Option]
Set the minimum of the scale relative to the "-dataelt1dcol scal=" data.
Possible values: `any`. Default value: `data minimum`.

`-dataelt1dscalemax` *`real`* [Option]
Set the maximum of the scale relative to the "-dataelt1dcol scal=" data.
Possible values: `any`. Default value: `data maximum`.

`-dataelt1dscaleticks` *`real`* [Option]
Set the ticks of the scale relative to the "-dataelt1dcol scal=" data. Provide a string composed of values separated by commas. (Use '`_`' to get a blank space.)
Possible values: `any`. Default value: `none`.

`-dataelt1drad` *`char_string`* [Option]
Set the radii of the 1D element. See option '`-dataelt3dedgerad`' for the argument format.
Possible values: `any`. Default value: `proportional to 3D elset size`.

The following options enable to load data relative to the 0D mesh elements. Note that the options can be applied to element sets instead of elements by changing '`elt`' to '`elset`'.

`-dataelt0dcol` *`char_string`* [Option]
Set the colours of the 0D elements. See option '`-dataelt3dcol`' for the argument format.
Possible values: `any`. Default value: `black`.

`-dataelt0dcolscheme` *`char_string`* [Option]
Set the colour scheme used to get colours from the data of the 0D elements loaded with option '`-dataelt0dcol`'. See option '`-dataelt3dcolscheme`' for the argument format.
Possible values: `see option '-dataelt3dcolscheme'`. Default value: `see option '-dataelt3dcolscheme'`.

`-dataelt0dscalemin` *`real`* [Option]
Set the minimum of the scale relative to the "-dataelt0dcol scal=" data.
Possible values: `any`. Default value: `data minimum`.

`-dataelt0dscalemax` *`real`* [Option]
Set the maximum of the scale relative to the "-dataelt0dcol scal=" data.
Possible values: `any`. Default value: `data maximum`.

`-dataelt0dscaleticks` *`real`* [Option]
Set the ticks of the scale relative to the "-dataelt0dcol scal=" data. Provide a string composed of values separated by commas. (Use '`_`' to get a blank space.)
Possible values: `any`. Default value: `none`.

`-dataelt0drad` *`char_string`* [Option]
Set the radii of the 0D element. See option '`-dataelt3dedgerad`' for the argument format.
Possible values: `any`. Default value: `proportional to 3D elset size`.

The following options enable to load data relative to the nodes.

`-datanodecoo` *`char_string`* [Option]
Set the coordinates of the nodes. The argument can be the name of a file containing a list of coordinates, or a string indicating how the coordinates can be obtained. The string has the format '*`var`*`=file_name`', where '*`var`*' can be '`disp`' for displacements,, and '`file_name`' is the name of the file containing the data.
Possible values: `any`. Default value: `none`.

`-datanodecoofact` *`real`* [Option]
Set the value of the scaling factor to apply to the displacements of the nodes.
Possible values: `any`. Default value: `1`.

`-datanodecol` *file_name* [Option]
Set the colours of the nodes. See option '`-dataelt3dcol`' for the argument format.
Possible values: `any`. Default value: `black`.

`-datanodecolscheme` *char_string* [Option]
Set the colour scheme used to get colours from the data of the nodes loaded with option '`-datanodecol`'. See option '`-dataelt3dcolscheme`' for the argument format.
Possible values: `see option '-dataelt3dcolscheme'`. Default value: `see option '-dataelt3dcolscheme'`.

`-datanodescalemin` *real* [Option]
Set the minimum of the scale relative to the "-datanodecol scal=" data.
Possible values: `any`. Default value: `data minimum`.

`-datanodescalemax` *real* [Option]
Set the maximum of the scale relative to the "-datanodecol scal=" data.
Possible values: `any`. Default value: `data maximum`.

`-datanodescaleticks` *real* [Option]
Set the ticks of the scale relative to the "-datanodecol scal=" data. Provide a string composed of values separated by commas. (Use '`_`' to get a blank space.)
Possible values: `any`. Default value: `none`.

`-datanoderad` *file_name* [Option]
Set the radii of the nodes. See option '`-dataeltedgerad`' for the argument format.
Possible values: `any`. Default value: `none`.

The following options enable to colour the elements from data defined at the nodes.

`-datanode2eltcol` *file_name* [Option]
Set the colours of the elements from values defined at the nodes. See option '`-dataelt3dcol`' for the argument format.
Possible values: `any`. Default value: `black`.

`-datanode2eltcolscheme` *char_string* [Option]
Set the colour scheme used to get colours from the data loaded with option '`-datanode2eltcol`'. See option '`-dataelt3dcolscheme`' for the argument format.
Possible values: `see option '-dataelt3dcolscheme'`. Default value: `see option '-dataelt3dcolscheme'`.

`-datanode2eltscalemin` *real* [Option]
Set the minimum of the scale relative to the "-datanode2eltcol scal=" data.
Possible values: `any`. Default value: `data minimum`.

`-datanode2eltscalemax` *real* [Option]
Set the maximum of the scale relative to the "-datanode2eltcol scal=" data.
Possible values: `any`. Default value: `data maximum`.

`-datanode2eltscaleticks` *real* [Option]
Set the ticks of the scale relative to the "-datanode2eltcol scal=" data. Provide a string composed of values separated by commas. (Use '`_`' to get a blank space.)
Possible values: `any`. Default value: `none`.

6.1.4 Slice Settings

`-slicemesh` *`char_string`* [Option]

Use this option to plot one (or several) slice(s) of the mesh. Provide as argument the equation of the plane, under the form `{x,y,z}=value` (combine with commas). This option works for free meshes only (tetrahedral elements).
Possible values: `any`. Default value: `none`.

6.1.5 Show Settings

The following options apply to the full tessellation or mesh.

`-showtess` *`logical`* [Option]

Use this option to show or hide the tessellation.
Possible values: `0 or 1`. Default value: `1 if tess loaded and no mesh`.

`-showmesh` *`logical`* [Option]

Use this option to show or hide the mesh.
Possible values: `0 or 1`. Default value: `1 if mesh loaded and no slice`.

The following options apply to the entities of the tessellation.

`-showpoly` *`char_string`* [Option]

Specify the polyhedra to show. The argument can be: '`all`' for all, '*`file_name`*' to load polyhedron identifiers from a file, or any expression based on the following arguments: *id*, *cenx*, *ceny*, *cenz*, *volume*, *true*, *body*, and *faceqty*.
Possible values: `any`. Default value: `all if tess loaded`.

`-showface` *`char_string`* [Option]

Specify the faces to show. The argument can be: '`all`' for all, '`@`*`file_name`*' to load face identifiers from a file, or any expression based on the following arguments: *id*, *cenx*, *ceny*, *cenz*, *area*, *ff*, *true*, *body*, *verqty*, *edgeqty* and *poly_shown*.
Possible values: `any`. Default value: `none`.

`-showedge` *`char_string`* [Option]

Specify the edges to show. The argument can be: '`all`' for all, '`@`*`file_name`*' to load edge numbers from a file, or any expression based on the following arguments: *id*, *cenx*, *ceny*, *cenz*, *length*, *true*, *body*, *face_shown*, *poly_shown* and *cyl*. The *cyl* variable is useful to hide "fake" edges which would appear within the faces of the grains which are on the circular part of a cylinder. Use argument `cyl==0` to hide them.
Possible values: `any`. Default value: `none`.

`-showver` *`char_string`* [Option]

Specify the vertices to show. The argument can be: '`all`' for all, '`@`*`file_name`*' to load vertex numbers from a file, or any expression based on the following arguments: *id*, *cenx*, *ceny*, *cenz*, *true*, *body*, *edge_shown*, *face_shown* and *poly_shown*.
Possible values: `any`. Default value: `none`.

`-showfaceinter` *`logical`* [Secondary option]

Show the interpolations of the tessellation faces (if any).
Possible values: `0 or 1`. Default value: `0`.

The following options apply to the entities of the mesh.

`-showelt` *char_string* [Option]
Specify the elements to show. The argument can be: '`all`' for all , '`@file_name`' to load element numbers from a file, or any expression based on the following arguments: *id*, *cenx*, *ceny*, *cenz*, *volume*, *elset_true*, *elset_body* and *elset_id*.
Possible values: `any`. Default value: `all if mesh loaded (and no tessellation) and nothing -show'd`.

`-showelset` *char_string* [Option]
Specify element sets to show. Refer to option '`-showpoly`' for the available arguments.
Possible values: `any`. Default value: `none`.

`-showelt1d` *char_string* [Option]
Specify the 1D elements to show. The argument can be: '`all`' for all , '`@file_name`' to load element numbers from a file, or any expression based on the following arguments: *cenx*, *ceny*, *cenz*, *length*, *elset_true*, *elset_body*, *id*, *elt3d_shown* and *cyl*. The *cyl* variable is useful to hide "fake" 1-D elements which would appear within the faces of the grains which are on the circular part of a cylinder. Use argument `cyl==0` to hide them.
Possible values: `any`. Default value: `none`.

`-showshadow` *logical* [Option]
Show the shadows. If you want "true" colours, switch this option off.
Possible values: `0 or 1`. Default value: `1`.

6.1.6 Camera Settings

`-cameracoo[,x,y,z]` *char_string* [Option]
Specify the camera coordinates. The expression can be based on the following arguments: *tesscentre*, *meshcentre*, *v* and *cameralookat*.
Possible values: `any`. Default value: `cameralookat+v`.

`-cameralookat[,x,y,z]` *char_string* [Option]
Specify the point the camera looks at. The expression can be based on the following arguments: *O* (the origin), *tesscentre* and *meshcentre*.
Possible values: `any`. Default value: `tesscentre if tess printed, meshcentre if mesh printed`.

`-cameraangle` *real* [Option]
Specify the opening angle of the camera along the horizontal direction (degrees).
Possible values: `any`. Default value: `25`.

`-cameraprojection` *char_string* [Option]
Specify the type of projection of the camera.
Possible values: `perspective or orthographic`. Default value: `perspective`.

6.1.7 Output Image Settings

`-imagesize` *integer integer* [Option]
Specify the width and height of the image (in pixels).
Possible values: `any any`. Default value: `1200 900`.

`-imagebackground` *real real real* [Option]
Specify the colour of the background (normed RGB levels).
Possible values: `any`. Default value: `1 1 1`.

`-imageantialias` *integer* [Option]
Use antialiasing to produce a smoother image.
Possible values: `any (consider 1 to 3)`. Default value: `0`.

`-imageformat` *`string`* [Option]
Specify the format of the output image.
Possible values: `png or pov`. Default value: `png`.

6.1.8 Scripting

`-loop char_string real real real ... -endloop` [Option]
Use this option to make a loop. Provide as argument the name of the loop variable, its initial value, the loop increment value, the final value, then the commands to execute. An example of use of the -loop / -endloop capability is provided in the Examples Section.
Possible values: `any`. Default value: `none`.

6.1.9 Advanced Options

`-includepov` *`char_string`* [Option]
Use this option to include additional objects to the figure, under the form of a POV-Ray file. Provide as argument the name of the POV-Ray file.
Possible values: `any`. Default value: `none`.

6.2 Output Files

The output files are:

- Image file, '`.png`': a bitmapped image (the alpha channel is off).
- POV-Ray file, '`.pov`': a POV-Ray script file.

A PNG image can be obtained from the '`.pov`' file by invoquing POV-Ray as follows (see the POV-Ray documentation for details and further commands):

```
$ povray Input_File_Name=file.pov +Wimage_width +Himage_height -D .
```

6.3 Examples

Below are some examples of use of neper -VS. Illustrations can be found at `http://neper.sourceforge.net/neper_vs.html`.

1. Print out tessellation n100-id1.tess as image 'img.png'.
   ```
   $ neper -VS -loadtess n100-id1.tess -print img
   ```
2. Print out tessellation n100-id1.tess with a camera angle of 12 degrees, an image size of 400x400 pixels and an image antialiasing level of 2 (better quality).
   ```
   $ neper -VS -loadtess n100-id1.tess -cameraangle 12 -imagesize 400
     400 -imageantialias 2 -print img
   ```
3. Print out tessellation n100-id1.tess with the colours written in file 'n100-id1.col' to render the grains.
   ```
   $ neper -VS -loadtess n100-id1.tess -datapolycol n100-id1.col
     -cameraangle 12 -imagesize 400 400 -imageantialias 2 -print img
   ```
4. Print out tessellation n100-id1.tess with the colours written in file 'n100-id1.col' to render the grains and nicely shown vertices and edges.
   ```
   $ neper -VS -loadtess n100-id1.tess -datapolycol n100-id1.col
     -dataverrad 0.03 -dataedgerad 0.015 -datavercol red -dataedgecol
     0:90:180 -cameraangle 12 -imagesize 400 400 -imageantialias 2
     -print img
   ```

5. Print out tessellation n100-id1.tess with the grains coloured according to their orientations (orientation file: 'n100-id1.ori').

```
$ neper -VS -loadtess n100-id1.tess -datapolycol
  ori=n100-id1.ori -cameraangle 12 -imagesize 400 400
  -imageantialias 2 -print img
```

6. Print out mesh n100-id1.msh as image 'img.png'. Set the radius of the element edges to 0.0015 and the radius of the 1D element edges to 0.0045.

```
$ neper -VS -loadmesh n100-id1.msh -dataeltedgerad 0.0015
  -dataelt1drad 0.0045 -cameraangle 12 -imagesize 400 400
  -imageantialias 2 -print img
```

7. Print out mesh n100-id1.msh with the colours written in file 'n100-id1.col' to render the grains.

```
$ neper -VS -loadmesh n100-id1.msh -dataelsetcol
  ori=n100-id1.ori -dataeltedgerad 0.0015 -dataelt1drad 0.0045
  -cameraangle 12 -imagesize 400 400 -imageantialias 2 -print img
```

8. Print out mesh n100-id1.msh, but only the inner grains.

```
$ neper -VS -loadmesh n100-id1.msh -dataelsetcol ori=n100-id1.ori
  -showelset "body>0" -dataeltedgerad 0.0015 -dataelt1drad 0.0045
  -cameraangle 12 -imagesize 400 400 -imageantialias 2 -print img
```

9. Print out mesh n100-id1.msh, but only the elements with z<0.5.

```
$ neper -VS -loadmesh n100-id1.msh -dataelsetcol ori=n100-id1.ori
  -showelt "cenz<0.5" -dataeltedgerad 0.0015 -dataelt1drad 0.0045
  -cameraangle 12 -imagesize 400 400 -imageantialias 2 -print img
```

10. Print out 3 slices of mesh n100-id1.msh.

```
$ neper -VS -loadmesh n100-id1.msh -dataelsetcol ori=n100-id1.ori
  -slicemesh "x=0.5,y=0.5,z=0.5" -cameraangle 12 -imagesize 400
  400 -imageantialias 2 -print img
```

Appendix A File Formats

A.1 Tessellation file '.tess'

Here are details on the '.tess' file format version 1.10. The developers may note that read and write functions are available as `neut_geo_fscanf ()` and `neut_geo_fprintf ()`, defined in directories '`neut/neut_geo/neut_geo_fscanf`' and '`neut/neut_geo/neut_geo_fprintf`'.

```
***tess
 **format
   format
 **general
   n id type morphology
 **vertex
   total_number_of_vertices
   ver_id  dom_entity_type dom_entity_id
           number_of_edges edge_1 edge_2 ...
           ver_x ver_y ver_z ver_state
   ...
 **edge
   total_number_of_edges
   edge_id dom_entity_type dom_entity_id
           ver_1 ver_2
           number_of_faces face_1 face_2 ...
           edge_state
   ...
 **face
   total_number_of_faces
   face_id dom_entity_type dom_entity_id
           poly_1 poly_2
           face_eq_a face_eq_b face_eq_c face_eq_d
           number_of_vertices ver_1   ver_2   ...
                              edge_1* edge_2* ... (As many edges as vertices.)
           face_state face_point face_point_x face_point_y face_point_z
           face_ff
   ...
 **polyhedron
   total_number_of_polyhedra
   poly_id poly_centre_x poly_centre_y poly_centre_z
           poly_true poly_body
           number_of_faces face_1* face_2* ...
   ...
 **domain
  *general
   dom_type
  *vertex
   total_number_of_dom_vertices
   dom_ver_id  ver_1
               number_of_edges dom_edge_1 dom_edge_2 ...
   ...
  *edge
```

```
  total_number_of_dom_edges
  dom_edge_id number_of_dom_tess_edges edge_1 edge_2 ...
              dom_ver_1 dom_ver_2
              dom_face_1 dom_face_2
  ...
 *face
  total_number_of_dom_faces
  dom_face_id number_of_dom_tess_faces dom_tess_face_1 dom_tess_face_2 ...
              dom_face_eq_a dom_face_eq_b dom_face_eq_c dom_face_eq_d
              number_of_dom_vertices dom_ver_1  dom_ver_2  ...
                                     dom_edge_1 dom_edge_2 ...
  ...
***end
```

where (with identifiers being integer numbers)

- *format* is the file format, currently '1.10' (character string).
- *n* is the number of polyhedra of the tessellation (option '-n').
- *id* is the identifier of the tessellation (option '-id').
- *type* is the type of tessellation (always 'standard').
- *morphology* is a character string indicating the morphology of the tessellation or polyhedra (option '-morpho'). It equals to 'poisson' for a Poisson Voronoi tessellation, 'tocta' for truncated octahedra, 'columnar[x,y,z]' for columnar grains, 'bamboo[x,y,z]' for bamboo grains or 'custom' for user-specified coordinates of the polyhedron germs (use of option '-centrecoo').
- *total_number_of_vertices* is the total number of vertices.
- *dom_entity_type* is an integer equal to the type of domain entity that a tessellation vertex, edge or face belongs to. The value is equal to 0 for a domain vertex, 1 for a domain edge and 2 for a domain face, and -1 for none.
- *dom_entity_id* is the identifier of a domain entity that a tessellation vertex, edge or face belongs to. The domain entity can be a domain vertex, edge or face, as indicated by *dom_entity_type*. If *dom_entity_type* = -1, *dom_entity_id* carries on no information and has a value of 0.
- *ver_id* is the identifier of a vertex and ranges from 1 to total_number_of_vertices.
- *number_of_edges* is the number of edges a vertex belongs to.
- *edge_1*, *edge_2*, ... are identifiers of edges.
- *edge_1**, *edge_2**, ... are identifiers of the edges of a face, signed according to their orientation in the face.
- *ver_x*, *ver_y* and ver_z are the three coordinates of a vertex (real numbers).
- *ver_state* is an integer indicating the state of a vertex. For a standard tessellation (no regularization), it equals 0. For a regularized tessellation, it equals 0 if the vertex has not been modified by regularization and is higher than 0 otherwise.
- *total_number_of_edges* is the total number of edges.
- *edge_id* is the identifier of an edge and ranges from 1 to total_number_of_edges.
- *ver_1*, *ver_2*, ... are identifiers of vertices.
- *number_of_faces* is the number of faces an edge belongs to, or to the number of faces a polyhedron has, depending on the context.
- *face_1*, *face_2*, ... are identifiers of faces.

- *face_1**, *face_2**, ... are identifiers of the faces of a polyhedron, signed according to their orientations in the polyhedron (positive if the normal of the face is pointing outwards and negative if it is pointing inwards).
- *edge_state* is an integer indicating the state of an edge (always 0).
- *total_number_of_faces* is the total number of faces.
- *face_id* is the identifier of a face and ranges from 1 to total_number_of_faces.
- *poly_1* and *poly_2* are identifiers of polyhedra. *poly2* is negative if the face belongs to a domain face, thereby having only one parent polyhedron In that case, its value is negative *dom_entity_id*.
- *face_eq_a*, *face_eq_b*, *face_eq_c* and *face_eq_d* are the parameters of the equation of a face: $face\_eq\_a\,x + face\_eq\_b\,y + face\_eq\_c\,z = face\_eq\_d$. The parameters are scaled so that $face\_eq\_a^2 + face\_eq\_b^2 + face\_eq\_c^2 = 1$.
- *number_of_vertices* is the number of vertices that a face has.
- *face_state* is an integer indicating the state of a face. For a standard tessellation (no regularization), it equals 0. For a regularized tessellation, it equals 0 if it has not been modified by regularization and 1 otherwise.
- *face_point* is an integer indicating the point used for the interpolation of a face. For a standard tessellation (no regularization), it equals 0. For a regularized tessellation: if the point is the face barycentre, it equals 0; if the point is one of the face vertices, it equals to the position of the vertex in the list of vertices of the face (the list being: ver_1 ver_2 ...). It equals -1 if the point is undefined.
- *face_point_x*, *face_point_y* and *face_point_z* are the coordinates of the point used for the interpolation of a face (equal 0 if undefined).
- *face_ff* is a real value equal to the "flatness fault" of a face. For a standard tessellation, it equals 0. For a regularized tessellation, it is the maximum angle between the normals at two points of a face, expressed in degrees.
- *total_number_of_polyhedra* is the total number of polyhedra.
- *poly_id* is the identifier of a polyhedron and ranges from 1 to total_number_of_polyhedra.
- *poly_centre_x*, *poly_centre_y* and *poly_centre_z* are the coordinates of the centre of a polyhedron.
- *poly_true* is an integer equal to the "true" value of a polyhedron (see Chapter 3 [Module -FM], page 13).
- *poly_body* is an integer equal to the "body" value of a polyhedron (see Chapter 3 [Module -FM], page 13).
- *dom_type* is the type of the domain (one of cube, cylinder, poly and planes).
- *total_number_of_dom_vertices* is the total number of domain vertices.
- *dom_ver_id* is the identifier of a domain vertex and ranges between 1 to total_number_of_dom_vertices.
- *total_number_of_dom_edges* is the total number of domain edges.
- *dom_edge_id* is the identifier of a domain edge and ranges between 1 to total_number_of_dom_edges.
- *number_of_dom_tess_edges* is the number of tessellation edges that a domain edge has.
- *dom_ver_1*, *dom_ver_2*, ... are identifiers of the domain vertices that a domain edge or face has.
- *dom_face_1* and *dom_face_2* are identifiers of the domain faces that a domain edge has.
- *total_number_of_dom_faces* is the total number of domain faces.

- *dom_face_id* is the identifier of a domain face and ranges from `1` to `total_number_of_dom_faces`.
- *number_of_dom_tess_faces* is the number of tessellation faces that a domain face has.
- *dom_tess_face_1*, *dom_tess_face_2*, `...` are the identifiers of the tessellation faces that a domain face has.
- *dom_face_eq_a*, *dom_face_eq_b*, *dom_face_eq_c* and *dom_face_eq_d* are the parameters of the equation of a domain face and are defined in the same way than *face_eq_a*, etc. (see above).
- *number_of_dom_vertices* is the number of domain vertices (and edges) that a domain face has.
- *dom_edge_1*, *dom_edge_2*, `...` are identifiers of the domain edges that a domain face has.

Appendix B Mathematical and Logical Expressions

B.1 Mathematical Expressions

Neper can handle mathematical expressions. It makes use of the GNU `libmatheval` library. The expression must contain no space, tabulation or new-line characters, and match the following syntax[1]:

Supported constants are (names that should be used are given in parenthesis): e (`e`), log2(e) (`log2e`), log10(e) (`log10e`), ln(2) (`ln2`), ln(10) (`ln10`), pi (`pi`), pi / 2 (`pi_2`), pi / 4 (`pi_4`), 1 / pi (`1_pi`), 2 / pi (`2_pi`), 2 / sqrt(pi) (`2_sqrtpi`), sqrt(2) (`sqrt`) and sqrt(1 / 2) (`sqrt1_2`).

Variable name is any combination of alphanumericals and `_` characters beginning with a non-digit that is not elementary function name.

Supported elementary functions are (names that should be used are given in parenthesis): exponential (`exp`), logarithmic (`log`), square root (`sqrt`), sine (`sin`), cosine (`cos`), tangent (`tan`), cotangent (`cot`), secant (`sec`), cosecant (`csc`), inverse sine (`asin`), inverse cosine (`acos`), inverse tangent (`atan`), inverse cotangent (`acot`), inverse secant (`asec`), inverse cosecant (`acsc`), hyperbolic sine (`sinh`), cosine (`cosh`), hyperbolic tangent (`tanh`), hyperbolic cotangent (`coth`), hyperbolic secant (`sech`), hyperbolic cosecant (`csch`), hyperbolic inverse sine (`asinh`), hyperbolic inverse cosine (`acosh`), hyperbolic inverse tangent (`atanh`), hyperbolic inverse cotangent (`acoth`), hyperbolic inverse secant (`asech`), hyperbolic inverse cosecant (`acsch`), absolute value (`abs`), Heaviside step function (`step`) with value 1 defined for x = 0, Dirac delta function with infinity (`delta`) and not-a-number (`nandelta`) values defined for x = 0, and error function (`erf`).

Supported unary operation is unary minus (`'-'`).

Supported binary operations are addition (`'+'`), subtraction (`'+'`), multiplication (`'*'`), division multiplication (`'/'`) and exponentiation (`'^'`).

Usual mathematical rules regarding operation precedence apply. Parenthesis (`'('` and `')'`) could be used to change priority order.

Neper includes additional functions: the minimum function (`min(a,b)`) and the maximum function (`max(a,b)`). *a* and *b* can be any expression following the above-described syntax. Moreover, square brackets (`'['` and `']'`) and curly brackets (`'{'` and `'}'`) can be used instead of the parentheses.

B.2 Logical Expressions

The logical operators supported are: = (`==`), ≠ (`!=`), ≥ (`>=`), ≤ (`<=`), > (`>`), < (`<`), AND (`&&`) and OR (`||`).

[1] Taken from the `libmatheval` documentation.

Appendix C Colours

Colours can be specified in two ways: by name or by RGB channel values, as detailed in the following.

Here is the list of the available colours (character string and RGB channel values):
aliceblue (240, 248, 255), antiquewhite (250, 235, 215), aqua (0, 255, 255), aquamarine (127, 255, 212), azure (240, 255, 255), beige (245, 245, 220), bisque (255, 228, 196), black (0, 0, 0), blanchedalmond (255, 235, 205), blue (0, 0, 255), blueviolet (138, 43, 226), brown (165, 42, 42), burlywood (222, 184, 135), cadetblue (95, 158, 160), chartreuse (127, 255, 0), chocolate (210, 105, 30), coral (255, 127, 80), cornflowerblue (100, 149, 237), cornsilk (255, 248, 220), crimson (220, 20, 60), cyan (0, 255, 255), darkblue (0, 0, 139), darkcyan (0, 139, 139), darkgoldenrod (184, 134, 11), darkgray (64, 64, 64), darkgreen (0, 100, 0), darkgrey (64, 64, 64), darkkhaki (189, 183, 107), darkmagenta (139, 0, 139), darkolivegreen (85, 107, 47), darkorange (255, 140, 0), darkorchid (153, 50, 204), darkred (139, 0, 0), darksalmon (233, 150, 122), darkseagreen (143, 188, 143), darkslateblue (72, 61, 139), darkslategray (47, 79, 79), darkslategrey (47, 79, 79), darkturquoise (0, 206, 209), darkviolet (148, 0, 211), deeppink (255, 20, 147), deepskyblue (0, 191, 255), dimgray (105, 105, 105), dimgrey (105, 105, 105), dodgerblue (30, 144, 255), firebrick (178, 34, 34), floralwhite (255, 250, 240), forestgreen (34, 139, 34), fuchsia (255, 0, 255), gainsboro (220, 220, 220), ghostwhite (248, 248, 255), gold (255, 215, 0), goldenrod (218, 165, 32), gray (128, 128, 128), grey (128, 128, 128), green (0, 255, 0), greenyellow (173, 255, 47), honeydew (240, 255, 240), hotpink (255, 105, 180), indianred (205, 92, 92), indigo (75, 0, 130), ivory (255, 255, 240), khaki (240, 230, 140), lavender (230, 230, 250), lavenderblush (255, 240, 245), lawngreen (124, 252, 0), lemonchiffon (255, 250, 205), lightblue (173, 216, 230), lightcoral (240, 128, 128), lightcyan (224, 255, 255), lightgoldenrodyellow (250, 250, 210), lightgray (211, 211, 211), lightgreen (144, 238, 144), lightgrey (211, 211, 211), lightpink (255, 182, 193), lightsalmon (255, 160, 122), lightseagreen (32, 178, 170), lightskyblue (135, 206, 250), lightslategray (119, 136, 153), lightslategrey (119, 136, 153), lightsteelblue (176, 196, 222), lightyellow (255, 255, 224), lime (0, 255, 0), limegreen (50, 205, 50), linen (250, 240, 230), magenta (255, 0, 255), maroon (128, 0, 0), mediumaquamarine (102, 205, 170), mediumblue (0, 0, 205), mediumorchid (186, 85, 211), mediumpurple (147, 112, 219), mediumseagreen (60, 179, 113), mediumslateblue (123, 104, 238), mediumspringgreen (0, 250, 154), mediumturquoise (72, 209, 204), mediumvioletred (199, 21, 133), midnightblue (25, 25, 112), mintcream (245, 255, 250), mistyrose (255, 228, 225), moccasin (255, 228, 181), navajowhite (255, 222, 173), navy (0, 0, 128), oldlace (253, 245, 230), olive (128, 128, 0), olivedrab (107, 142, 35), orange (255, 165, 0), orangered (255, 69, 0), orchid (218, 112, 214), palegoldenrod (238, 232, 170), palegreen (152, 251, 152), paleturquoise (175, 238, 238), palevioletred (219, 112, 147), papayawhip (255, 239, 213), peachpuff (255, 218, 185), peru (205, 133, 63), pink (255, 192, 203), plum (221, 160, 221), powderblue (176, 224, 230), purple (128, 0, 128), red (255, 0, 0), rosybrown (188, 143, 143), royalblue (65, 105, 225), saddlebrown (139, 69, 19), salmon (250, 128, 114), sandybrown (244, 164, 96), seagreen (46, 139, 87), seashell (255, 245, 238), sienna (160, 82, 45), silver (192, 192, 192), skyblue (135, 206, 235), slateblue (106, 90, 205), slategray (112, 128, 144), slategrey (112, 128, 144), snow (255, 250, 250), springgreen (0, 255, 127), steelblue (70, 130, 180), tan (210, 180, 140), teal (0, 128, 128), thistle (216, 191, 216), tomato (255, 99, 71), turquoise (64, 224, 208), violet (238, 130, 238), wheat (245, 222, 179), white (255, 255, 255), whitesmoke (245, 245, 245), yellow (255, 255, 0), yellowgreen (154, 205, 50).

Any other colour of known RGB channel values can be defined by forming a character string of format: '`R_value,G_value,B_value`'. The '`,`' separator can be changed to one of '`/`', '`.`', '`:`' and '`;`'.

Appendix D Versions

```
New in 1.10.0 (June 2012):
- General: New (hopefully simpler) installation procedure based on Cmake.
  Added support for domains of any convex polyhedral shape.
- module -VS: major code rewriting and option changes. New capabilities
  for defining the colours and sizes of the tessellation / mesh
  (including gradients).  Added options to show only specific parts of the
  tessellation / mesh and to view slices of a mesh. Other small
  enhancements.
- module -T : added option ‘-domain’ to specify the shape of the domain
  (cuboidal, cylindrical or of any convex shape), small bug fixes, added
  centroid Voronoi tessellation generation (option -centroid), merged
  option -centrecoo into option -morpho, added polyhedron centroid
  coordinates in file .stt3, changed option -load to -loadtess, added
  output format ‘.ply’ (thanks Ehsan!).
- module -FM: mesh partitionning needs libscotch version 5.1.12 or later,
  small bug fixes, changed default value of -faset to "" (i.e. no faset
  in output), fixed bug for Abaqus output, added polyhedron centroid
  coordinates in file .stt3, added output format ‘.ply’ (geometry only).
- module -MM: new options -dsize and -scale, new option -loadmesh, new
  option -outdim, changed arguments of -ttype, changed default value of
  -faset to "" (i.e. no faset in output), fixed bug for Abaqus output,
  small bug fixes.

New in 1.9.2 (September 2011):
- module -T: added option -morpho for specifying the type of grain
  structure (equiaxed, columnar or bamboo), merged option -regular
  with -morpho, added post-processing -neighbour option for information
  on the polyhedron neighbours, added geo (Gmsh geometry) output format
  (mostly for visualization), fixed bugs.
- module -MM: proper processing of the input tess files, added msh
  (Gmsh) and inp (Abaqus) output formats, added options -morpho and
  -centrecoo (as in module -T), small bug fixes, code cleaning.
- module -FM: added geo (Gmsh geometry) output format (mostly for
  visualization), small bug fixes.
- documentation: small corrections.

New in 1.9.1 (May 2011):
- module -FM: fixed bug occurring when -mesh3dalgo is not set by the
  user.  Small other bug fixes.
- module -MM: small bug fixes.

New in 1.9.0 (Apr 2011):
This is a major release.  Neper now has its own paper:
"R.Quey, P.R. Dawson and F. Barbe.  Large-scale 3D random polycrystal
 for the finite element method: Generation, meshing and remeshing.
 Computer Methods in Applied Mechanics and Engineering,
 Vol. 200, pp. 1729--1745, 2011."
Please cite it in your works if you use Neper.
- General: added option --rcfile to disregard / change the
  initialization file; big distribution and source clean up; bug fixes.
```

```
- module -T: added capability to generate regular morphologies
  (truncated octahedra), tess file format bumped to 1.9; big clean up.
- module -FM: included multimeshing, remeshing and mesh partitioning
  capabilities; big clean up.  Neper now uses the *standard* Gmsh
  distribution for 2D and 3D meshings (versions >=2.4.2).  Strongly
  reduced memory usage.
- module -O: added capability to handle different orientation
  descriptors.
- module -VS: new visualization module to generate publication-quality
  images (PNG format) of the tessellations, meshes and more...

New in 1.8.1 (Aug 2009):
- upgraded website at http://neper.sourceforge.net
- module -T: new file format ***tess1.8, new option -restart to load an
  existing tessellation (not through std input any more), new option
  -printformat, bug fixes.
- module -MM: bug fixes.
- module -FM: new output format mae, new option -restart to restart from
  an existing geometry or mesh (options -mesh and -conv removed); new
  options -printformat and -maeextension; better mesh numbering (+ new
  options -elementfirstid and -nodefirstid), new way to choose the node
  sets to output (-nset 4), fixed option -estat, renamed -bwcy-clmin to
  -clmin, cleaned bunch of options, bug fixes.
- module -O: added option -euleranglesconvention (Bunge, Roe & Kocks);
  new output formats mae and geof (option -format).
- manual: some corrections.

New in 1.8.0 (Jul 2009):
- First GPL-distributed version of Neper.
```

Appendix E GNU General Public License

GNU General Public License

Version 3, 29 June 2007

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program—to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

   "This License" refers to version 3 of the GNU General Public License.

   "Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

   "The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

   To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

   A "covered work" means either the unmodified Program or a work based on the Program.

   To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a devel copy. Propagation includes copying, distribution (with or without modification), making available to the distrib, and in some countries other activities as well.

   To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

   An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

   The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

   A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

   The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the distrib in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

   The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

   All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

   You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

   Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

   No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

   When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

   You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

   You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

   You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

   a. The work must carry prominent notices stating that you modified it, and giving a relevant date.
   b. The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c. You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
d. If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a. Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
b. Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
c. Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
d. Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
e. Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general distrib at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is distribly documented (and with an implementation available to the distrib in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

   "Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

   When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

   Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

   a. Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b. Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
c. Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
d. Limiting the use for distribity purposes of names of licensors or authors of the material; or
e. Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
f. Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

    Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

    An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

    You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

    A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

    A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

    Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

    In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

    If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a distribly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

    If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

    If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

    Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

    The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

    Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

    If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's distrib statement of acceptance of a version permanently authorizes you to choose that version for the Program.

    Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

    THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRIT-

ING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

    IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

    If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the distrib, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
one line to give the program's name and a brief idea of what it does.
Copyright (C) year name of author

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or (at
your option) any later version.

This program is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see http://www.gnu.org/licenses/.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
program Copyright (C) year name of author
```

```
This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type 'show c' for details.
```

The hypothetical commands '`show w`' and '`show c`' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see `http://www.gnu.org/licenses/`.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read `http://www.gnu.org/philosophy/why-not-lgpl.html`.