NePeR Reference Manual

The documentation for NePeR 1.8
A software to generate 3D random polycrystals for the finite element method.

12 July 2009

Romain Quey

Table of Contents

Copying Conditions **1**

1 Introduction **3**

1.1 What is NePeR 3
1.2 The History of NePeR 4
1.3 Installing NePeR 4
1.3.1 Particular Requirements 4
1.3.2 Installation 4
1.4 Getting Started 5
1.4.1 General Commands 5
1.4.2 Call a Module 5
1.4.3 Initialization File 5
1.4.4 Miscellaneous 6

2 NePeR Capabilities **7**

2.1 Voronoi Tessellation Generation 7
2.1.1 Voronoi Tessellation 7
2.1.2 Types of Tessellation 7
2.1.3 Domain Size, Tessellation Deformation 8
2.2 Voronoi Tessellations Mapped Meshing 9
2.3 Voronoi Tessellations Free Meshing 10
2.3.1 Meshing Parameters 10
2.3.2 Small Edge Deletion 11
2.3.3 Small Angles 13
2.3.4 Influence of the Geometry Modification on Mechanical Fields 13
2.4 Crystallographic Orientations Generation 15

3 Tessellation Generation: neper -T **17**

3.1 Module Commands 17
3.1.1 Input Data 17
3.1.2 General Options 17
3.1.3 Tessellation Options 17
3.1.4 Deformation Options 18
3.1.5 Output Options 18
3.1.6 Printing Options 18
3.1.7 Other capabilities 18
3.2 Result Files 18
3.3 Examples 19

4 Tessellation Mapped Meshing: neper -MM **21**

4.1 Module Commands 21
4.1.1 Input Data 21
4.1.2 General Options 21
4.1.3 Tessellation Options 21
4.1.4 Mesh Options 22
4.1.5 Output Options 22
4.2 Result Files 22
4.3 Examples 23

5 Tessellation Free Meshing: neper -FM 25
5.1 Module Commands 25
5.1.1 Input Data 25
5.1.2 General Options 25
5.1.3 General Meshing Options 25
5.1.4 Small Edges Deletion Options 26
5.1.5 Small Angles Meshing Options 26
5.1.6 Output Options 27
5.1.7 Other Options 28
5.2 Printing Options 28
5.3 Other Capabilities 28
5.4 Result Files 28
5.5 Examples 29

6 Crystallographic Orientation Generation: neper -O 31
6.1 Module Commands 31
6.1.1 Input Data 31
6.1.2 General Options 31
6.2 Result Files 31
6.3 Examples 31

7 Tips and Tricks – Versions 33
7.1 Tips and Tricks 33
7.2 Versions 33

Appendix A GNU General Public License 35

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 A [GNU General Public License], page 35). Further information about this license is available from the GNU Project webpage `http://www.gnu.org/copyleft/gpl-faq.html`.

NePeR can be downloaded from `http://neper.sourceforge.net`. As the software evolves (new functionalities, bug fixes, manual improvements...), you should check this site to get the latest version. You should also consider subscribing to the 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`.
- neper-users: the list for users.

  Please send all questions, bug reports (including any data enabling to reproduce it), requests or any errors or omissions in this manual to this list. Feel free; feedback is very welcome.

  To subscribe, visit `https://lists.sourceforge.net/lists/listinfo/neper-users`, to send a message, use `neper-users@lists.sourceforge.net`.

If you use NePeR, please cite it in your work. The recommended way is,
NePeR: A software to build 3D random polycrystals for the finite element method (version 1.8), `http://neper.sourceforge.net`.
Also, I would like very much to hear from you directly. Please send an e-mail to `rquey@users.sourceforge.net` describing how you use NePeR. This will motivate further developments of the software that I hope will benefit you.

1 Introduction

This reference manual is made of six chapters. This chapter aims at introducing briefly the **neper** program. The second chapter describes the program capabilities. The next chapters are devoted to describing the modules of the program. Each chapter describes exhaustively a module: all the available commands are listed into detail, the result files are described and some examples provided.

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`'.

Unfortunately, this document may contain mistakes, and the program itself may contain unknown bugs. Please send any comment, request, suggestion or bug report (including any data enabling to reproduce it) to the author (`rquey@users.sourceforge.net`).

1.1 What is NePeR

neper is a (16000 code lines) software to generate 3D random polycrystals for the finite element method. The polycrystal are represented as 3D Voronoi tessellations. **neper** also enables to simulate diffusive phase transformation of polycrystals, but this capability is not distributed yet.

neper is built around four modules. A brief description of the modules is given hereafter.

- **neper -T** is devoted to generating Voronoi tessellations. Any parallelepipedic domain can be tessellated, and the germs of the Voronoi cells are randomly distributed. Tessellations can also be deformed after being built.
- **neper -MM** enables to build a mapped mesh of a tessellation (i.e. using brick elements). Mapped meshes of standard tessellations, periodic tessellations and so-called *subdomain type* tessellations can be created. Options for phase transformation simulations are also available.
- **neper -FM** aims at building a free mesh of a tessellation. Optimized mesh rules are implemented into **neper**; furthermore, it is possible to modify the tessellation geometry (that is called *small edge deletion* in the **neper**'s jargon) to improve mesh quality. To control tessellation modification and mesh quality, some general statistics are available.
- **neper -O** is dedicated to generating "random" crystallographic orientations for the tessellation polyhedra.

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

1.2 The History of NePeR

This project was started february 2003 during the engineer studies of Romain Quey, at the Department of Mechanics, INSA de Rouen, France (http://www.insa-rouen.fr), under supervision of Dr. Fabrice Barbe. Since july 2003, the project has been developed on spare time.

1.3 Installing NePeR

1.3.1 Particular Requirements

neper is written in ANSI C and can be compiled and run under any Unix-like system. You will need a C compiler (e.g. the GNU compiler `gcc`), as well as the GSL and the Gmsh program:

- the GSL library

  It can be downloaded from the webpage http://www.gnu.org/software/gsl/, and installed by following the instructions given in the reference manual.

- the Gmsh program (homepage: http://www.geuz.org/gmsh/)

  In fact, **neper** requires a slightly modified version of Gmsh version 1.60.1 that is included into this distribution of **neper**. To be compiled, Gmsh firstly requires the GSL (see above). The usual version of the program also requires FLTK (http://www.fltk.org, version 1.1.x); but a non-graphical version – which is sufficient for **neper** and does not require FLTK – may also be built.

  To install Gmsh with the Graphical User Interface (GUI) (GSL and FLTK required):

  ```
  $ ./configure; make; make install;
  ```

  To install Gmsh without GUI (sufficient for **neper** and only GSL is required):

  ```
  $ ./configure --disable-gui; make; make install;
  ```

Moreover, **neper** provides images at the Asymptote format `.asy`. Asymptote is a powerful vector graphics language, see http://asymptote.sourceforge.net .

1.3.2 Installation

According to your local configuration, you may need to tune the '`Makefile`' file prior to the compilation. Namely, if your GSL installation directory location is not '`/usr/local/include/`', it has to be specified as the `GSLLIB` variable at top of the '`Makefile`' file. The location can be obtained thanks to the command '`gsl-config --cflags | sed -e "s/-I//g"`'. For example, the specification could be: `GSLLIB=/opt`.

Next, **neper** can be compiled as follows:

```
$ make;
```

and

```
$ make install;
```

to copy the binary file to the standard system location ('`/usr/local/bin/`').

Provided that the `info` program is properly installed at your site, you may want to install the **neper** info documentation. This can be achieved by

1. copying file '`neper.info`' into your info directory[1];
2. issuing the command (this may be useless)

   ```
   $ install-info info_directory/neper.info info_directory/dir
   ```

[1] usually '`/usr/info`' or '`/usr/local/info`' or '`/usr/doc/info`'

1.4 Getting Started

1.4.1 General Commands

First, **neper** includes some general commands:

- to get a short help:

  ```
  $ neper --help
  ```

- to get the program version:

  ```
  $ neper --version
  ```

Provided that the info manual is correctly installed at your site, the following command:

```
$ info neper
```

should give you access to the on-line reference manual.

1.4.2 Call a Module

The modules of **neper** can simply be run as follows:

```
$ neper module_name module_arguments
```

where *module_arguments* can include both required input data and options. They can be given in arbitrary order and are to be specified as follows: "*option_name option_value*" (excepted for input data file names). Logical options can be selected by giving the value '`1`', or disabled by giving '`0`'.

Among all the options, note that those beginning by '`-bwcy`' are for backward compatibility only (you should use them only to disable some good functionalities of the program.).

1.4.3 Initialization File

When a module of **neper** is run, it looks for an initialization file to load. This file is called '`.neperrc`' and must be located in your home directory. If the initialization file is found, the module looks for the occurrence of '`neper module_name`' in it and reads any argument until the next occurrence of '`neper`' (which denotes 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 classically they are limited to defining any frequently-used options.

An example of initialization file is given below:

```
neper comments -------------------------------------------------------
This is my default initialization file (~/.neperrc).

neper -T -v 0

neper -MM -v 0 -morder 2

neper -FM -v 1 -morder 2 -maxff 10 -estat 1 -gmsh my-gmsh-path
-geo2geob 1 -rcl 0.8

neper -O -v 0

neper comments -------------------------------------------------------
```

If '`neper module_name`' is not found, or if the initialization file is not found in the home directory, **neper** will just consider the command line arguments. Finally, note that if an argument is initialized several times (e.g. once in the initialization file and once in the command line), the last specified value is retained.

1.4.4 Miscellaneous

- String completion is available for all the program arguments, so they may be abbreviated as long as the abbreviation is not ambiguous.
- For some reasons, it is not recommended to have several runs of **neper** at the same time *and* at the same location. Of course, you can have several runs of **neper** successively at the same location or simultaneously at different locations ...
- Don't create a Gmsh configuration file (`~/.gmshrc`) against a run of **neper -FM**. This would occur while working with Gmsh, and cause **neper** to abort.

2 NePeR Capabilities

In **neper**, a polycrystalline medium is modelized as a Voronoi tessellation. The tessellations can be meshed either regularly (cubic elements) or freely (tetrahedral elements). **neper** can also generates random crystallographic orientations for the grains. In addition, **neper** includes capabilities to simulate a diffusive phase transformation of the medium from particular sites (e.g. the quadruple points), and following chosen kinetics laws.

2.1 Voronoi Tessellation Generation

2.1.1 Voronoi Tessellation

A *tessellation* of a *n*-D space is a collection of *n*-D entities that fills the space with no overlaps and no gaps. These entities, called Voronoi *cells* (or *polyhedra*) are formally defined as zones of influence of a particular set of points, corresponding to their *germs* (of *centres*).

Being given

- a spatial domain: $D \in \Re^3$,
- a set of points: $E = \{A_i(\underline{x}_i)\}$, randomly distributed into D,
- an euclidean norm: $d(\bullet, \bullet)$,

every point A_i is associated a Voronoi cell C_i as follows:

$$C_i = \{P(\underline{x}) \in D \mid d(P, A_i) < d(P, A_j)\ \forall j \neq i\}$$

A Voronoi cell is a convex polyhedron; the intersection of two Voronoi cells is a plane, called *tessellation face*; the intersection of three Voronoi cells is a straight edge, called *tessellation edge*; and the intersection of four Voronoi cells is a point, called *tessellation vertex*. No more than four polyhedra can intersect. In a polycrystal, these entities correspond to the *grain*, *grain boundary*, *triple line* and *quadruple point*, respectively.

In **neper**, the domain is taken as a finite parallelepiped (even though any convex domain could be tessellated).

2.1.2 Types of Tessellation

In **neper**, three *types* of tessellation can be built. Only the first one is available in **neper -T** at the moment, but the two others are available through the mapped meshing technique (see Chapter 4 [Module -MM], page 21).

- *Standard tessellation*

  All the germs are situated inside the domain.
- *Periodic tessellation*

  The germs are not only inside the domain: the set of germs is periodically repeated in the 3 directions of space. Hence, the tessellation is periodic.
- *Subdomain type tessellation*

  The tessellation is like picked up from a larger tessellated domain which can be considered to be infinite (D_∞).

The actual number of cells within the domain can vary according to the type of tessellation. As for the standard tessellations, it is obvious that the number of cells is equal to the specified number of germs, n. As for the periodic tessellations, the number of cells also is equal to n, but the cells at the domain boundary are discontinuous. At opposite, the subdomain type tessellations generally does not contain n polyhedra: n actually stands for the mean number of germs per domain volume that applies to (D_∞). According to the position of the domain within

(D_∞), the number of germs it contains vary. The domain can also contain polyhedra whose germs are around the domain.

The type of tessellation actually does not influence the results of FE calculations (e.g. effective properties predictions) significantly. At opposite, it does much more for phase transformations, that can occur, e.g., from quadruple points. In that case, the subdomain type tessellation are to be used.

2.1.3 Domain Size, Tessellation Deformation

In **neper**, it is possible to choose the size of the domain, but also to deform the tessellation once it has been built. While a cubic domain may be used for RVE calculation, a "flat" domain may be useful for modelling, e.g., a recrystallized thin sheet of metal. On the other hand, tessellation deformation may be useful to account for morphological texture of a material. See Figure 2.1 for illustrations of these three configurations.

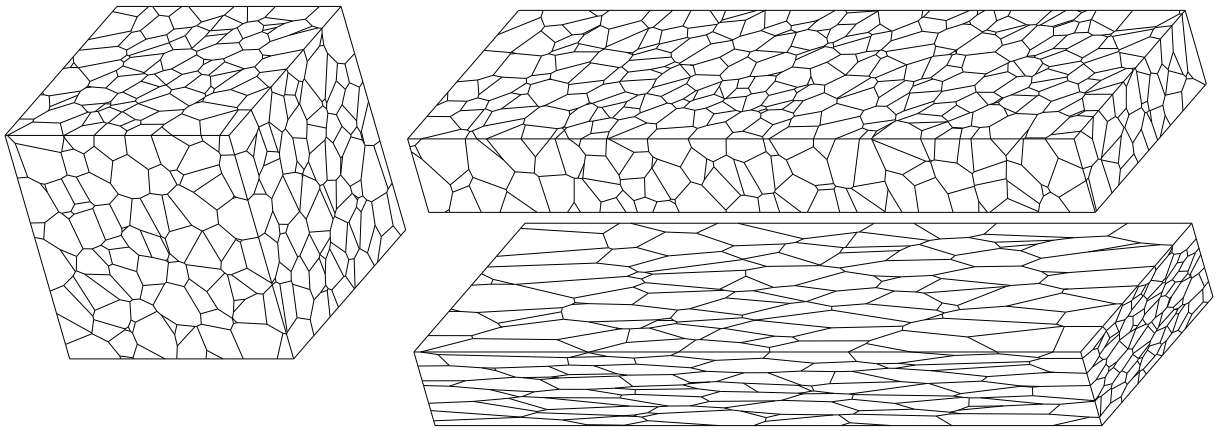

Figure 2.1: Examples of standard tessellations with 1000 cells. (left) Tessellation of a cubic domain, (top right) Tessellation of a "flat" domain, (bottom right) Deformed tessellation of a cubic domain.

Examples of NePeR commands:

- Generate a standard tessellation:
  `neper -T -n 1000 -id 1`
- The same, but specify the domain size:
  `neper -T -n 1000 -id 1 -dsize 3 0.33 1`
- The same, but apply tessellation deformation:
  `neper -T -n 1000 -id 1 -morpho 3 0.33 1`

2.2 Voronoi Tessellations Mapped Meshing

The *mapped meshing* technique uses regular cubic elements. So, the actual tessellation differs from the initial one: intersections between cells are poorly described. Nevertheless, this simplified approach makes it possible to generate quite easily periodic and subdomain type tessellations (see Figure 2.2). Here, the characteristic mesh parameter is the number of elements along an edge of the cube.

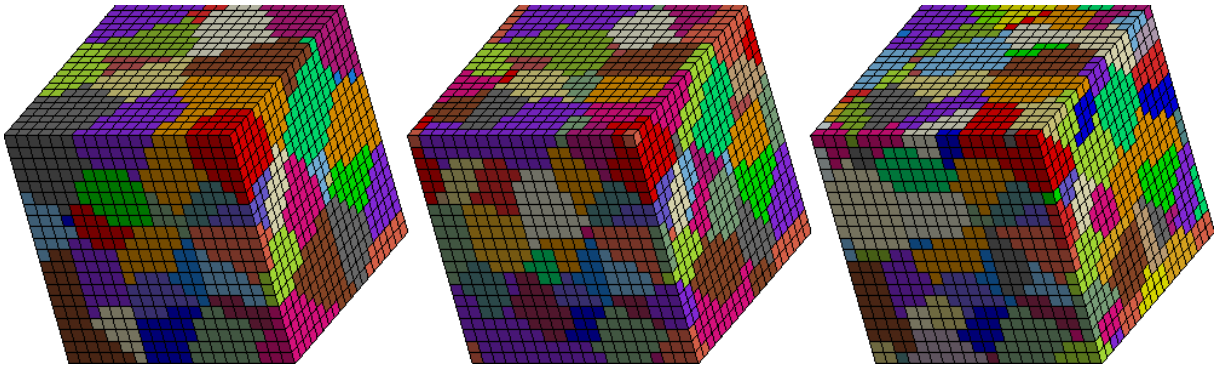

Figure 2.2: Mapped meshes of the three available types of tessellation. (left) Standard tessellation, (middle) Periodic tessellation. (right) Subdomain type tessellation.

Examples of NePeR commands:

- Mesh a standard tessellation and specify the mesh parameters:
  `neper -MM -n 100 -id 1 -msize 40 -morder 2`
- The same, but for a periodic tessellation:
  `neper -MM -n 100 -id 1 -msize 40 -morder 2 -ttype 1`
- The same, but for a subdomain type tessellation:
  `neper -MM -n 100 -id 1 -msize 40 -morder 2 -ttype 2`

2.3 Voronoi Tessellations Free Meshing

The *free meshing* technique uses tetrahedral elements. This makes it possible to describe exactly the tessellation. However, free meshing of Voronoi tessellations is particularly difficult because of the random feature of the geometry and the numerous geometrical details, e.g. the small tessellation edges (see Figure 2.3). As for the first point, one have to rely on a robust, automatic meshing software: the free mesher Gmsh from Christophe Geuzaine, including Triangle from Jonathan Shewchuk and Netgen from Joachim Schoberl, is a good candidate. As for the second point, it is obvious that, to avoid element distortion, the mesh has to be refined on every geometrical detail.
This section firstly presents the mesh rules that are implemented in **neper** to ensure mesh quality as far as possible. Then, the *small edge deletion* is presented, which is an approach really particular to **neper** consisting in deleting some of the geometrical details. Finally, one focuses on the other difficulty when meshing Voronoi tessellation: the small angles of the geometry.

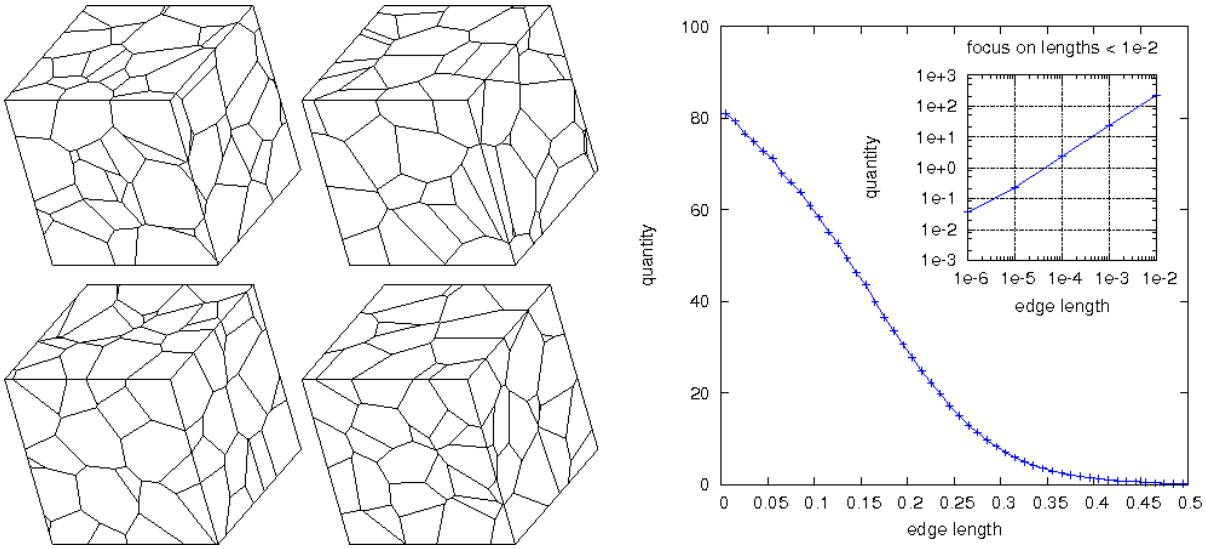

Figure 2.3: (left) 4 different 100 cells tessellations of a cubic domain, (right) Their average edge length distribution, showing a large number of very small edges.

2.3.1 Meshing Parameters

To understand how the meshing process is controlled, let us remind that (at least for the Delaunay triangulation techniques that we use), the mesh is built in a bottom-up flow by successively creating the 0-D mesh (of the tessellation vertices), the 1-D mesh (of the tessellation edges), the 2D mesh (of the tessellation faces) and the 3D mesh (of the tessellation polyhedra). Each n-D meshing step is performed while taking into account the (n-1)-D mesh constraints. Moreover, the mesh size at dimension n is extrapolated from the one at dimension n-1.

In **neper**, two main parameters enable to control the meshing process. They directly apply on the 0-D and 1-D meshes and, as explained before, indirectly on the 2D and 3D meshes.
The first parameter is the *standard characteristic length* (`cl`) of the elements, i.e. the characteristic length that would apply to every element if there were no mesh constraints (a relative parameter, `rcl`, is defined as `cl` / average_edge_length). The second parameter is the *progression factor* (`pcl`) for the transition between the characteristic length of the elements at the tessellation vertices and the one of the edge interior elements. (At the edge extremities (i.e. at its vertices), the element characteristic lengths is about the length of the smaller vertex edge if there is a small edge, and equal to the standard characteristic length else.) The influence of the meshing parameters is illustrated on Figure 2.4. Compare figures (left) and (middle)

for `cl` and figures (left) and (right) for `pcl`. Figures (left) and (middle) exhibit strong mesh over-refinements near the small edges of the geometry – there are about as many elements in the refined zones than in the non-refined zones, but mesh quality is ensured. In fact, *the smaller the edge, the smaller its influence on the microstructure morphology, but the strong the mesh over-refinement*. At the contrary, figure (right) exhibits a homogeneous element characteristic length, but there are bad quality elements near the tessellation small edges – such a high `pcl` value is to be avoided.

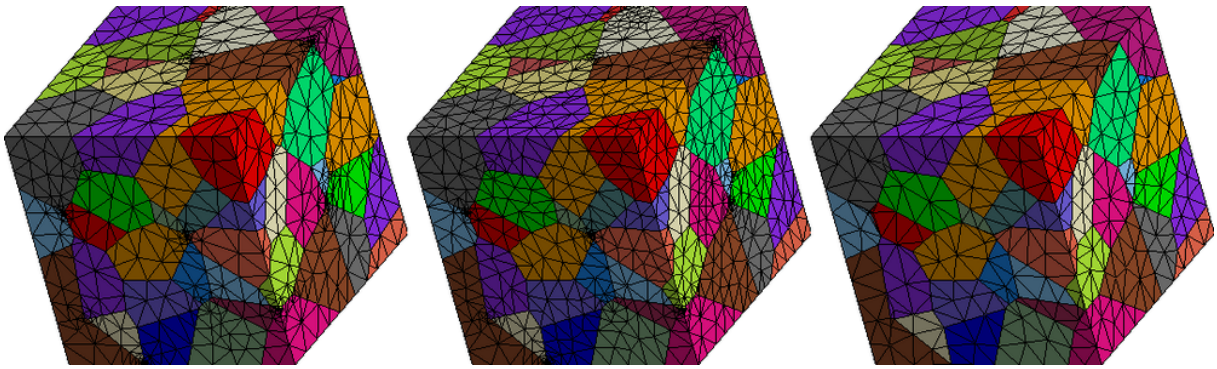

Figure 2.4: Influence of the meshing parameters: `rcl` and `pcl`. (left) `rcl = 1.2`, `pcl = 1.5`; (middle) `rcl = 0.9`, `pcl = 1.5`; (right) `rcl = 1.2`, `pcl = 1000`.

2.3.2 Small Edge Deletion

In addition, the program is able to modify the tessellation geometry to avoid mesh refinement on the geometrical details such as the "small" tessellation edges: this is called *small edge deletion* in the **neper**'s jargon. In its classical definition, an edge is said to be *small* if it requires mesh refinement (i.e. if its length is lower than the element standard characteristic length divided by the progression factor). As a small edge is deleted and replaced by a single vertex, the neighbouring tessellation faces can become non-plane: *flatness faults* can appear (that is impossible for real Voronoi tessellations), see Figure 2.5. This requires to apply an interpolation method for the face interior. In **neper**, the faces are divided into triangular parts. Each triangular part generally join one face edge to the face centre. Interpolation actually applies on the face mesh but not on the geometry, which makes it possible to avoid new 1-D mesh constraints.

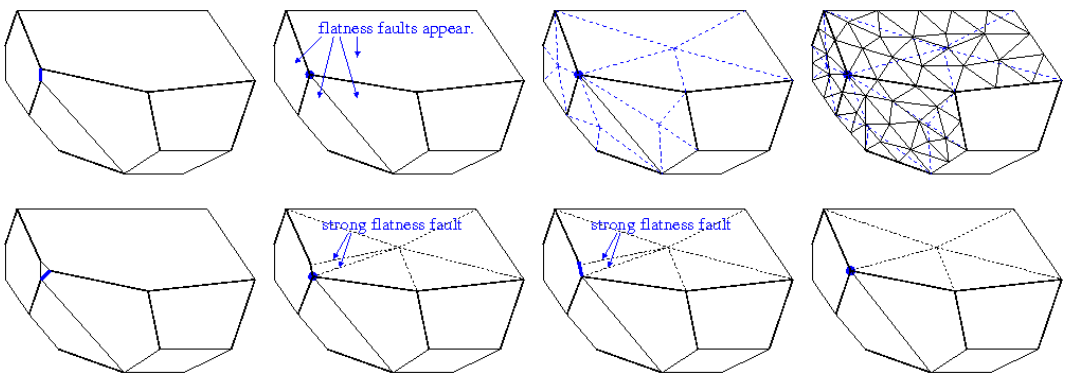

Figure 2.5: Main steps for a small edge deletion. (top) Standard case, (bottom) More problematic case where there are two neighbouring small edges.

The effect of the small edge deletion on the geometry is illustrated on Figure 2.6(top): the amount of small edges dramatically decreases and there is a distribution of flatness faults. Note

that, even if the maximum allowed value for the flatness faults is 10 degrees, most of them are far smaller. Furthermore, Figure 2.6(bottom) shows that the global morphology does not really change. The effect on the mesh is illustrated on Figure 2.7: while increasing the allowed flatness fault, the mesh size dramatically decreases – up to a factor of 2, while mesh quality is retained. This way, one can obtain a microstructure mesh with an homogeneous and quality mesh.

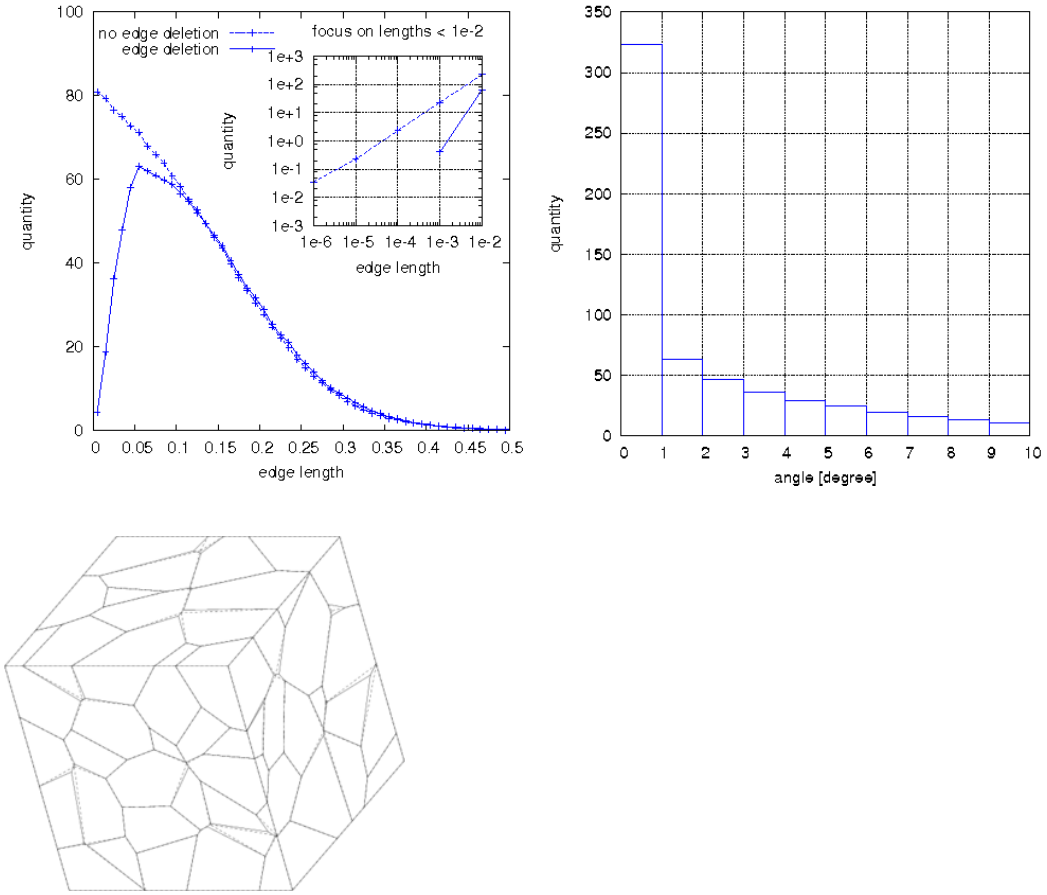

Figure 2.6: Effect of the small edge deletion (100 cells tessellations) on geometry. (top left) Edge length distributions, (top right) Flatness fault distributions, (bottom) Change in morphology (dash line = tessellation without edge length deletion),

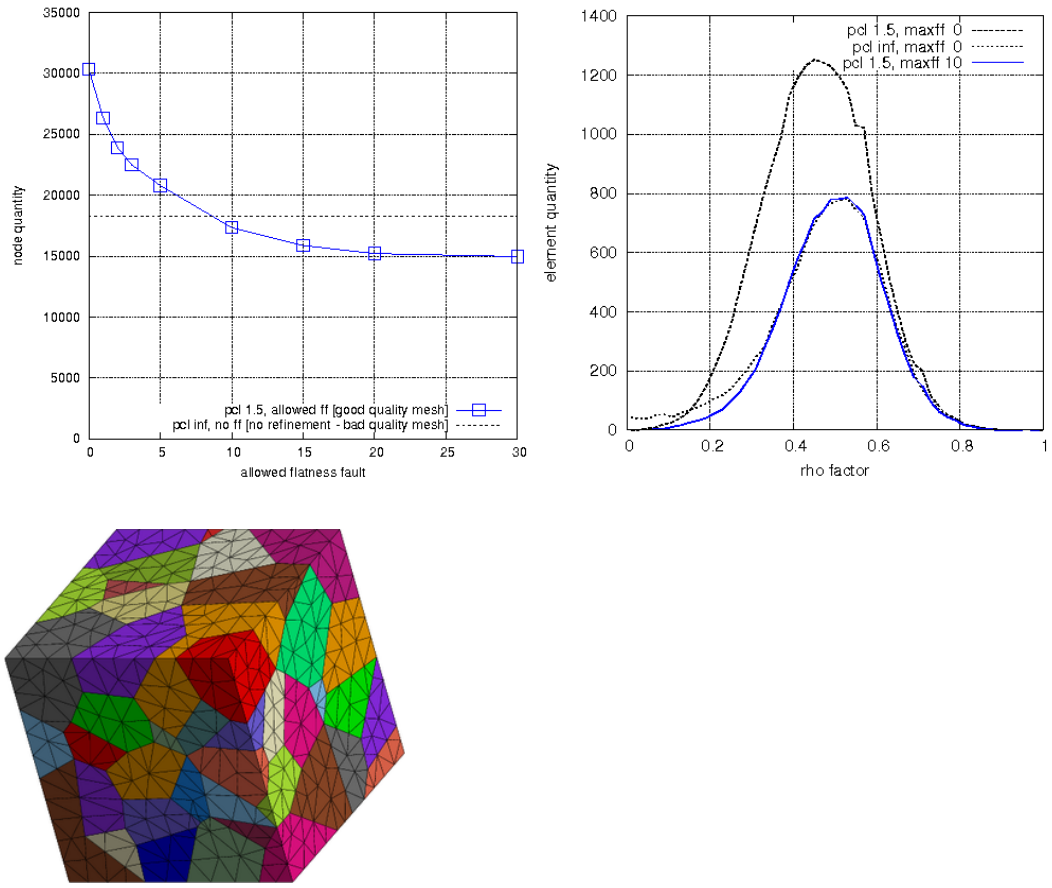

Figure 2.7: Effect of the small edge deletion (100 cells tessellations) on mesh. (top left) Mesh size vs maximum flatness fault, (top right) Distribution of element quality (rho factor = min element edge length / max element edge length), (bottom) Mesh view: there is no mesh constraint, while mesh quality is retained.

2.3.3 Small Angles

It also appears that one must pay a special attention on the small angles of the geometry. Indeed, while meshing the geometry, at least one bad quality element is systematically generated on every small angle of the geometry. Since they seem really hard to avoid or remove, it is chosen to refine the mesh on them, which makes it possible to decrease the size (and the influence) of the small bad quality elements. This is done by defining a maximum value for an angle to be "small" and an element characteristic length at these angles. Since there are not so much small angles, their mesh refinements does not really change the overall mesh size.

2.3.4 Influence of the Geometry Modification on Mechanical Fields

Figure 2.8 illustrates the effect of the small edge deletion on the mechanical fields provided by a finite elements calculation carried out with ZéBuLoN (`http://www.nwnumerics.com`). This is the case of a Zn alloy (transverse isotropic elasticity) submitted to tension; the output field is the equivalent VonMises stress. It appears that the geometry modification, while allowing to save computation-time, does not modify local stress fields.

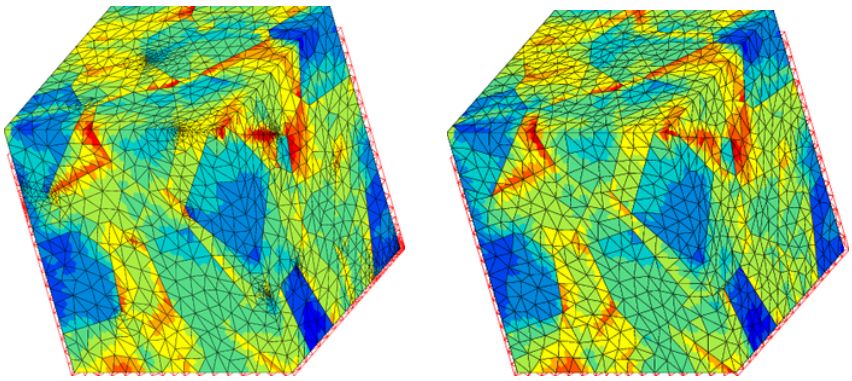

Figure 2.8: Effect of the small edge deletion on stress fields. (left) Unmodified tessellation, (right) Modified tessellation.

Examples of NePeR commands:

- Mesh a tessellation with specific mesh parameters:
  `neper -FM n100-id1.tess -rcl 0.8 -pcl 1.5`
- The same, but with small edge length deletion:
  `neper -FM n100-id1.tess -rcl 0.8 -pcl 1.5 -maxff 20`
- The same, and refine mesh on the small angles:
  `neper -FM n100-id1.tess -rcl 0.8 -pcl 1.5 -maxff 20 -sa 10 -sarcl 0.2`

2.4 Crystallographic Orientations Generation

In **neper**, it is possible to generate random crystallographic orientations for the grains. They are given under the form of Euler angles, following the Bunge convention. Figure 2.9 is an equal area projection of a random set of 1000 orientations.

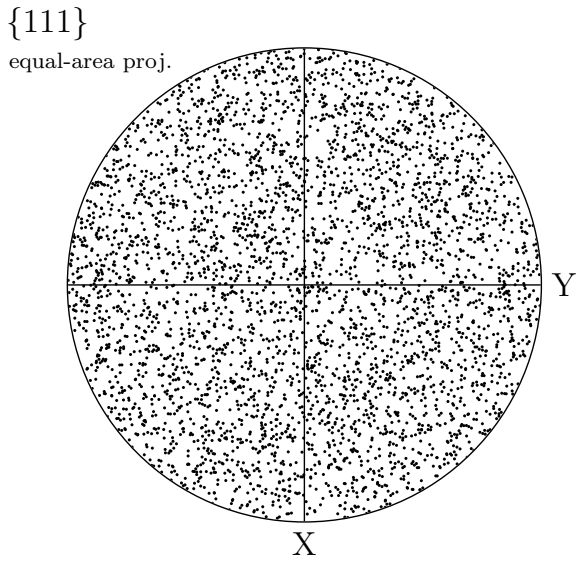

Figure 2.9: Equal-area projection of a set of 1000 orientations.

Examples of NePeR commands:

- Generate random crystallographic orientations:
  `neper -O -n 1000 -id 1`

3 Tessellation Generation: neper -T

neper -T is devoted to generating Voronoi tessellations. The domain can be any parallelepiped, and only standard tessellations can be built (see Chapter 4 [Module -MM], page 21 for mapped meshing of periodic and subdomain type tessellations). Tessellations can also be deformed after being built. The module mainly provides a tessellation file '`.tess`' containing an exhaustive description of the tessellation.

3.1 Module Commands

3.1.1 Input Data

The required input data are:

`-n` *integer*

Number of polyhedra of the tessellation.
Possible values: `1 to 1e7`. Default value: `none`.

and (`-id` and `-gcoo` are mutually exclusive),

`-id` *integer*

Identifier of the tessellation.
Possible values: `1 to 1e7`. Default value: `random`.

`-gcoo` *file_name*

Specify the germ coordinates. Give as argument the name of the file containing the 3 * n coordinates.
Possible values: `any`. Default value: `none`.

In the case where operations are to be done from an existing tessellation, you can give, as unique argument, the name of the tessellation file:

file_name

Name of the tessellation file.

3.1.2 General Options

`-o` *file_name*

Specify tessellation output file name.

`-v` *integer*

Set verbosity level.
Possible values: `0 to 1`. Default value: `0`.

3.1.3 Tessellation Options

`-dsize` *real real real*

Specify the domain size in the 3 directions of space.
Possible values: `any (positive)`. Default value: `1`.

`-ps2d` *integer*

Output a 2D tessellation (3D columnar in fact). Set the option to:
– `0` to do not use it;
– `1` to use it.
Possible values: `0 or 1`. Default value: `0`.

3.1.4 Deformation Options

`-deformation` *real real real*

Specify the x, y and z factors of which the tessellation is to be deformed after having been built.

Possible values: `any (positive)`. Default value: `none`.

3.1.5 Output Options

`-vcoo` *integer*

Provide the coordinates of the tessellation vertices together with the number of true Voronoi cells[1] they belong (0 to 4). Set the option to:
– `0` to do not use it;
– `1` to use it.

Possible values: `0 or 1`. Default value: `0`.
Result file: extension '`.vcoo`'.

3.1.6 Printing Options

`-print` *integer*

Print the geometry at the '`.mast`' and '`.asy`'[2] formats. Only the domain faces given with option '`printfaces`' are printed.
Give as argument:
– 0 to do not print;
– 1 to print specific domain faces (see option '`-printfaces`');
– 2 to print all the geometry.

Possible values: `0 to 2`. Default value: `0`.

`-printfaces` *integer ...*

Specify the domain faces to print. The number of the faces are: 1 and 2 for x- and x+, 3 and 4 for y- and y+, 5 and 6 for z- and z+. Give as argument the quantity of faces to print and their numbers.

Possible values: `[1-6] [1-6] ...`. Default value: `3 1 3 5`.

3.1.7 Other capabilities

`-pointpoly` *file_name*

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`'.

`-tesscentrepoly` *file_name*

This option is similar to '`-pointpoly`', but the points are taken as the centres of the tessellation described in the file given in argument (must be a valid '`.tess`' file).

Possible values: `any`. Default value: `none`.
Result file: extension '`-centre.polyid`'.

3.2 Result Files

The result files are:

1. tessellation file: '`.tess`'

   It contains an exhaustive description of the tessellation.

[1] An entity is said to be *true* if it is not influenced by the the domain boundary.

[2] `Asymptote` format; to get a postscript file, use `asy` *`file`*`.asy`.

2. vertex coordinates file: '`.vcoo`'

   It contains the list of all the tessellation vertices.

   Format: column 1: vertex id; column 2: nb of true polyhedra at which it belongs (0 to 4); columns 3 to 5: coordinates. The vertices are listed with ascending number of true polyhedra order.
3. polyhedron identifier file: '`.polyid`'

   It contains the identifiers of the polyhedra of which specific points belong (see options '`-pointpoly`' and '`-tesscentrepoly`'). By definition, they range from `1` to the maximum number of polyhedra in the tessellation. In the case where a point does not belong to any polyhedron (i.e. it is outside the domain), the returned identifier is `0`.

3.3 Examples

- Generate a tessellation in 1000 polyhedra, with identifier = 1

  ```
  $ neper -T -n 1000 -id 1
  ```
- The same, but consider a domain $(3 * 0.33 * 1)$ in size

  ```
  $ neper -T -n 1000 -id 1 -dsize 3 0.33 1
  ```
- The same, but deform the tessellation after building

  ```
  $ neper -T -n 1000 -id 1 -deformation 3 0.33 1
  ```
- The same, and print it as a mast file

  ```
  $ neper -T -n 1000 -id 1 -print 1
  ```

4 Tessellation Mapped Meshing: neper -MM

neper -MM mainly aims at building a mapped mesh (cubic elements) of a tessellation. Any standard tessellation provided by **neper -T** (file '`.tess`') can be meshed. **neper -MM** can also generate the tessellation mapped meshes as a single step (faster). At the present time, the domain must be a unit cube in that case, and it is also possible to build mapped meshes of periodic tessellations and subdomain type tessellations. The output is either an element set (elset) file '`.elset`' containing one elset per polyhedron, or the whole tessellation mapped mesh, written as a ZéBuLoN mesh file '`.geof`'. The mapped mesh itself (node and element definitions) can either be built by **neper** or loaded from an input file (which enables to save calculation time once the mesh has been created).

4.1 Module Commands

4.1.1 Input Data

The required input data are:

file_name

Name of the tessellation file.

or, the two following ones:

`-n` *integer*

Number of polyhedra of the tessellation (for standard and periodic tessellations), or Mean number of polyhedra per unit volume (for subdomain type tessellations).
Possible values: `1 to 1e7`. Default value: `none`.

`-id` *integer*

Identifier of the tessellation.
Possible values: `1 to 1e7`. Default value: `random`.

4.1.2 General Options

`-o` *file_name*

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

`-v` *integer*

Set verbosity level.
Possible values: `0 to 1`. Default value: `0`.

4.1.3 Tessellation Options

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

`-ttype` *integer*

Specify the type of tessellation. Set the option to:
– `0` to get a standard tessellation;
– `1` to get a periodic tessellation;
– `2` to get a "subdomain type" tessellation.
Possible values: `0 to 2`. Default value: `0`.

`-ps2d` *integer*

Provide a 2D tessellation (3D columnar in fact). Set the option to:
– `0` to do not use it;
– `1` to use it.
Possible values: `0 to 1`. Default value: `0`.

4.1.4 Mesh Options

`-msize` *integer*

Specify the mesh size (number of elements per unit length).
Possible values: `1 to 1e7`. Default value: `20`.

`-msize3` *integer integer integer*

Specify the mesh size (number of elements per unit length) along the X, Y and Z directions.
Possible values: `1 to 1e7 1 to 1e7 1 to 1e7`. Default value: `20 20 20`.

`-morder` *integer*

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

`-mesh` *integer*

Generate the mapped mesh (nodes and elements). Set the option to:
– `0` to do not use it;
– `1` to use it.
Possible values: `0 or 1`. Default value: `1`.

`-lmesh` *file_name*

Load a mapped mesh from a geof file.
Possible values: `any`. Default value: `none`.

4.1.5 Output Options

`-nset` *integer*

Provide nsets. Set the option to get nsets of:
– `0` nothing;
– `1` domain faces;
– `2` domain faces, corners and edges;
– `3` domain faces, corners and edges so as domain faces and edges "bodies".
Possible values: `0 to 3`. Default value: `1`.

`-faset` *integer*

Provide the fasets (one faset per face). Set the option to:
– `0` to do not use it;
– `1` to use it.
Possible values: `0 to 1`. Default value: `0`.

4.2 Result Files

As for the mesh, the result file is either an elset file '`.elset`' or a whole mesh file whether the mesh has been requested or not. An input file for the crystallographic orientations generation is also created. For the phase transformation simulation, a nucleation site file can be provided.

1. an elset file: '`.elset`'

   It contains the elsets corresponding to the polyhedra.

2. a mesh file: '`.geof`'

   It contains the whole mapped mesh of the tessellation (node and element definitions, nsets, elsets, and optionally fasets).
3. an input file for the crystallographic orientations generation: '`.oin`'

   This file must be used to generate the polyhedron crystallographic orientations with **neper -O**.

4.3 Examples

- Mesh a standard tessellation with a chosen mesh size

  ```
  $ neper -MM n100-id1.tess -msize 25
  $ neper -MM -n 100 -id 1 -msize 25 (faster)
  ```
- Mesh a standard tessellation with a chosen mesh size; import mesh from file

  ```
  $ neper -MM -n 100 -id 1 -msize 25 -lmesh My25SizedMesh.geof
  ```
- Mesh a standard tessellation with a chosen mesh size, just get the elsets

  ```
  $ neper -MM -n 100 -id 1 -msize 25 -mesh 0
  ```
- Mesh a periodic tessellation

  ```
  $ neper -MM -n 100 -id 1 -ttype 1
  ```
- Mesh a "subdomain type" tessellation

  ```
  $ neper -MM -n 100 -id 1 -ttype 2
  ```

5 Tessellation Free Meshing: neper -FM

neper -FM mainly aims at building a free mesh (tetrahedral elements) of a tessellation. The input data is a tessellation file ('`.tess`'). It is also possible to delete some of the small edges and/or to refine mesh near the small angles. The module provides a free mesh at the either at the Gmsh '`.msh`' format ar at the ZéBuLoN '`.geof`' format (`http://www.nwnumerics.com`). To control tessellation modification and mesh quality, some general statistics are also available.

5.1 Module Commands

5.1.1 Input Data

The required input data is:

file_name

Name of the tessellation file.

5.1.2 General Options

`-o` *file_name*

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

`-v` *integer*

Set verbosity level.
Possible values: `0 to 1`. Default value: `0`.

`-gmsh` *path_name*

Specify the access path for gmsh (if not the one given by the command '`which gmsh`').

5.1.3 General Meshing Options

`-cl or -rcl` *real*

Absolute or Relative Characteristic Length of the elements.
The Relative Characteristic Length is given relatively to the the overall tessellation edge length.
Possible values: `0 to 1e32`. Default value: `-rcl 1`.

`-pcl` *real*

Progression factor for the element characteristic lengths. Values of about 2 sound good.
Possible values: `1 (limit case) to 1e7`. Default value: `2`.

`-morder` *integer*

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

`-bwcy-clmin` *real*

Minimum characteristic length of the elements (not recommended).
Possible values: `0 to 1`. Default value: `none`.

`-bwcy-edgedis` *integer*

Meshing of the tessellation edges: interpolation method of the element length. Set the option to:
– `0` for a linear progression along the edge (like in Gmsh);
– `1` for a geometrical progression (with parameter '`-pcl`') to a limit element length ('`-cl`').
Possible values: `0 or 1`. Default value: `1`.

`-bwcy-meshimprovement2d` *integer*

Improve the 2D mesh by avoiding over-refinements *within* the faces (drawback of the default algorithm) by using another algorithm.
Possible values: `0 or 1`. Default value: `1`.

5.1.4 Small Edges Deletion Options

`-sel or -rsel` *real*

Absolute or Relative Small Edge (maximum) Length.
The Relative Small Edge Length is given relatively to the the overall tessellation edge length. By default the "small edge" maximal (absolute) length is equal to the standard characteristic length divided by the progression factor (this enables one to avoid mesh-refinements). With this option it is possible to choose a different length (take care to do not choose too large lengths).
Possible values: `0 to 1e30`. Default value: `-sel` *`cl/pcl`*.

`-maxff` *real*

Maximum face flatness fault (in degree) which is allowed.
As for the edge length distribution, values higher or equal to `10` sound quite good; whereas values of about `20` to `30` have a better influence on the geometry angle distribution.
Possible values: `0 to 180`. Default value: `0`.

`-mloop` *integer*

Number of loops of deletion of small edges to perform.
During each loop of deletion, the small edges are considered successively from the shortest to the largest. Use this option to choose how many loops to perform. `1` loop still leads to very satisfactory results. Use more to get better results. If one loop gives no edge deletion, the deletion process stops.
Possible values: `0 to 1e7`. Default value: `2`.

`-ffalgo` *integer*

Method of interpolation of the non-plane faces and choice of the method to measure the flatness fault (*ff*). The *geometrical ff* is the maximum angle between to triangular parts of the face; the *mesh ff* is the maximum angle between two elements. With the geometrical ff definition, the small edge deletion is faster and more reliable. Set the option to:
– `0` to get faces decomposed into boundary and plane central parts (Gmsh capability not recommended) / mesh ff;
– `1` to get faces interpolated into triangular parts / geometrical ff;
– `2` for faces interpolated into triangular parts / mesh ff.
Possible values: `0 to 2`. Default value: `2`.

`-bwcy-newver` *integer*

Specify the method to position a new vertex. Set the option to:
– `0` to use the barycentre of the two previous vertices;
– `1` to position it so as to minimize the distances to the initial face planes.
Possible values: `0 or 1`. Default value: `1`.

5.1.5 Small Angles Meshing Options

`-sa` *real*

Small Angle value (of the *geometry*).
Specify a maximum angle (degree) for an angle to be "small". According to the value of `sacl`, this makes it possible to refine the mesh on the small angles of the geometry.
Possible values: `0 to 180`. Default value: `0`.

`-sacl or -sarcl` *real*

Small Angle Absolute or Relative Characteristic Length of the element.
The Relative Characteristic Length is given relatively to the the overall tessellation edge length. Use this option to refine the mesh on the small angles of the geometry. This makes it possible to get smaller bad quality elements on the geometry small angles.
Possible values: `0 to 1e32`. Default value: `-sarcl` *rcl* `(i.e. no effect)`.

5.1.6 Output Options

`-format` *character_string*

Specify the format of output file(s). Here are the formats available: `geo`, `msh` and `geof`. Give as option argument the list of formats, separated by commas and without any space. Give as option argument the sum of the choices you want ('`4`' means just '`.geof`').
Possible values: `anyone of the above list`. Default value: `geof`.

`-nset` *integer*

Provide nsets. Set the option to get nsets of:
– `0` nothing;
– `1` domain faces;
– `2` domain faces corners and edges;
– `3` domain faces, corners and edges so as domain faces and edges "bodies".
Possible values: `0 to 3`. Default value: `1`.

`-liset` *integer*

Provide lisets. Set the option to get lisets of:
– `0` nothing;
– `1` domain faces.
Possible values: `0 or 1`. Default value: `0`.

`-edgel` *integer*

Provide the *geometry* edge lengths. Set the option to:
– `0` to do not use it;
– `1` to use it.
Possible values: `0 to 1`. Default value: `0`.
Result file: extension '`.edgel`'.

`-angle` *integer*

Provide the *geometry* face angles. Set the option to:
– `0` to do not use it;
– `1` to use it.
Possible values: `0 to 1`. Default value: `0`.
Result file: extension '`.angle`'.

`-estat` *integer*

Provide some statistics on the mesh elements: volume rho-factor and minimum angle between two edges. The rho-factor is defined as the ratio between lengths of the smallest and largest edges of the element. This option also provides a file containing all the element angles (12 angles per element). Set the option to:
– `0` to do not use it;
– `1` to use it.
Possible values: `0 to 1`. Default value: `0`.
Result file: extension '`.estat' and '.eangle`'.

`-nstat` *integer*

Provide some statistics on the mesh elements: valence. The valence is the number of edges per node. Set the option to:
– `0` to do not use it;
– `1` to use it.
Possible values: `0 to 1`. Default value: `0`.
Result file: extension '`.nstat`'.

5.1.7 Other Options

`-geo2geob` *integer*

Rename '`file.geo`' to '`file.geob`' when '`file.geof`' is created. (This is to avoid that the command '`Zmaster file.geof`' aborts when '`file.geo`' exists.) Set the option to:
– `0` to do not use it;
– `1` to use it.
Possible values: `0 to 1`. Default value: `0`.

5.2 Printing Options

`-print` *integer*

Print the geometry at the '`.mast`' and '`.asy`'[1] formats. Only the domain faces given with option '`printfaces`' are printed.
Give as argument:
– 0 to do not print;
– 1 to print.
Possible values: `0 or 1`. Default value: `0`.

`-printfaces` *integer ...*

Specify the domain faces to print. The number of the faces are: 1 and 2 for x- and x+, 3 and 4 for y- and y+, 5 and 6 for z- and z+. Give as argument the quantity of faces to print and their numbers.
Possible values: `[1-6] [1-6] ...`. Default value: `3 1 3 5`.

5.3 Other Capabilities

`-mesh` *file_name*

Specify a geometry file ('`.geo`') to be meshed.
Possible values: `any`. Default value: `none`.

`-convert` *file_name*

Specify a Gmsh mesh file ('`.msh`') to convert into a ZéBuLoN mesh file ('`.geof`') (file .fod is also needed).
Possible values: `any`. Default value: `none`.

5.4 Result Files

The available result files are:

1. a geometry file: '`.geo`'

   It contains the description of the geometry. It can be read by Gmsh.

2. a 'flatness fault' file: '`.ff`'

   It contains the list of the flatness faults of the non-plane faces (1st col.: face number, 2nd col.: face flatness fault).

[1] `Asymptote` format; to get a postscript file, use `asy` *`file`*`.asy`.

3. an 'edge length' file: '`.edgel`'

   It contains the lengths of all the tessellation edges.
4. an 'face angle' file: '`.angle`'

   It contains the angles of the faces (1st col.: angle id, 2nc col.: angle in degree).
5. a mesh file: '`.msh`'

   It contains the mesh of the tessellation. It can be read by Gmsh.
6. a mesh file: '`.geof`'

   It contains the mesh of the tessellation. It can be read by ZéBuLoN.
7. a boundary conditions file: '`.bc`'

   This file contains linear rellations between the displacements of some nodes of the domain, under the form of `*free` commands that are to be used into the `.inp` file.
8. an element statistics file: '`.estat`'

   It contains the element volumes rho-factors and minimum angle between two edges. (1st col.: element number, 2nd col.: volume, 3rd col.: rho-factor, 4th col.: min angle).
9. a node statistics file: '`.nstat`'

   It contains the node valences. Note that the nodes which are on the cube boundaries are not taken into account. (1st col.: node number, 2nd col.: node valence).

5.5 Examples

- Mesh a tessellation without edge deletion; just get the geof mesh

  ```
  $ neper -FM n100-id1.tess -rcl 0.8 -pcl 2 -format geof
  ```
- Mesh a tessellation with edge deletion; get all available output files; change the '`.geo`' file extension when '`.geof`' file is created

  ```
  $  neper -FM n100-id1.tess -maxff 10 -rcl 0.8 -pcl 2 -format geo,msh,geof
  -geo2geob 1 -edgel 1 -estat 1 -nstat 1
  ```
- Mesh a tessellation with edge deletion; specify the small edge length and refine the mesh on the geometry small angles

  ```
  $      neper -FM n100-id1.tess -maxff 10 -rcl 0.8 -pcl 2 -rsel 0.5 -sa 15
  -sarcl 0.1
  ```

6 Crystallographic Orientation Generation: neper -O

neper -O is dedicated to generating "random" crystallographic orientations. They are given by the Euler angles with the Bunge notation $(\varphi_1, \phi, \varphi_2)$.

The input data is either a file '`.oin`' (provided by **neper -MM**) or the data (`n`, `id`).

The module provides an orientation file '`.ori`'.

6.1 Module Commands

6.1.1 Input Data

The required input data are:

file.oin

Name of the input file.

or, the two following ones:

`-n` *integer*

Number of crystallographic orientations.
Possible values: `1 to 1e7`. Default value: `none`.

`-id` *integer*

Identifier of the set of orientations.
Possible values: `1 to 1e7`. Default value: `none`.

6.1.2 General Options

`-o` *file_name*

Specify orientation output file name.

`-v` *integer*

Set verbosity level.
Possible values: `0 to 1`. Default value: `0`.

6.2 Result Files

The result files are:

1. orientation file: '`.ori`'

   It contains the set of crystallographic orientations. Column 1: φ_1, column 2: ϕ, column 3: φ_2.

6.3 Examples

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

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

- Generate a set of crystallographic orientations from (`n`, `id`)

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

7 Tips and Tricks – Versions

7.1 Tips and Tricks

- Compiling Gmsh fails

  Run `make clean` and delete any remaining '`.o`' and '`.a`' files in the directory. Then, run `./configure --disable-gui; make` again.

- Meshing systematically fails for big meshes (typically > 1000 polyhedra)

  Check the value of `ulimit -n`. If it is close or lower than the number of polyhedra of the tessellation, you will have to increase this value: (as root) add `* hard nofile 1000000` in your '`/etc/security/limits.conf`' and `ulimit -n 1000000` in your '`/etc/profile`' and/or in your '`~/.bash_profile`' (for CSH, use `limit descriptors 1000000`).

7.2 Versions

```
# This is the VERSIONS file of NePeR.

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

Appendix A 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.