Gmsh

Gmsh Reference Manual

The documentation for Gmsh 2.7
A finite element mesh generator with built-in pre- and post-processing facilities

8 March 2013

Christophe Geuzaine
Jean-François Remacle

Short Contents

Table of Contents

Obtaining Gmsh

The source code and various pre-compiled versions of Gmsh (for Windows, Mac and Unix) can be downloaded from `http://geuz.org/gmsh/`. Gmsh is also directly available in pre-packaged form in various Linux and BSD distributions (Debian, Ubuntu, FreeBSD, ...).

If you use Gmsh, we would appreciate that you mention it in your work by citing the following paper: "C. 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, Volume 79, Issue 11, pages 1309-1331, 2009". A preprint of that paper as well as other references and the latest news about Gmsh development are available on `http://geuz.org/gmsh/`.

Copying conditions

Gmsh is “free software”; this means that everyone is free to use it and to redistribute it on a free basis. Gmsh 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 Gmsh that they might get from you.

Specifically, we want to make sure that you have the right to give away copies of Gmsh, that you receive source code or else can get it if you want it, that you can change Gmsh or use pieces of Gmsh 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 Gmsh, 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 Gmsh. If Gmsh 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 Gmsh are found in the General Public License that accompanies the source code (see Appendix G [License], page 225). Further information about this license is available from the GNU Project webpage `http://www.gnu.org/copyleft/gpl-faq.html`. Detailed copyright information can be found in Appendix F [Copyright and credits], page 221.

If you want to integrate parts of Gmsh into a closed-source software, or want to sell a modified closed-source version of Gmsh, you will need to obtain a different license. Please contact us directly for more information.

1 Overview

Gmsh is a three-dimensional finite element grid generator with a build-in CAD engine and post-processor. Its design goal is to provide a fast, light and user-friendly meshing tool with parametric input and advanced visualization capabilities.

Gmsh is built around four modules: geometry, mesh, solver and post-processing. All geometrical, mesh, solver and post-processing instructions are prescribed either interactively using the graphical user interface (GUI) or in text files using Gmsh's own scripting language. Interactive actions generate language bits in the input files, and vice versa. This makes it possible to automate all treatments, using loops, conditionals and external system calls. A brief description of the four modules is given hereafter.

1.1 Geometry: geometrical entity definition

Gmsh uses a boundary representation ("BRep") to describe geometries. Models are created in a bottom-up flow by successively defining points, oriented lines (line segments, circles, ellipses, splines, ...), oriented surfaces (plane surfaces, ruled surfaces, triangulated surfaces, ...) and volumes. Groups of geometrical entities (called "physical groups") can also be defined, based on these elementary geometric entities. Gmsh's scripting language allows all geometrical entities to be fully parametrized.

1.2 Mesh: finite element mesh generation

A finite element mesh is a tessellation of a given subset of the three-dimensional space by elementary geometrical elements of various shapes (in Gmsh's case: lines, triangles, quadrangles, tetrahedra, prisms, hexahedra and pyramids), arranged in such a way that if two of them intersect, they do so along a face, an edge or a node, and never otherwise. All the finite element meshes produced by Gmsh are considered as "unstructured", even if they were generated in a "structured" way (e.g., by extrusion). This implies that the elementary geometrical elements are defined only by an ordered list of their nodes but that no predefined order relation is assumed between any two elements.

The mesh generation is performed in the same bottom-up flow as the geometry creation: lines are discretized first; the mesh of the lines is then used to mesh the surfaces; then the mesh of the surfaces is used to mesh the volumes. In this process, the mesh of an entity is only constrained by the mesh of its boundary. For example, in three dimensions, the triangles discretizing a surface will be forced to be faces of tetrahedra in the final 3D mesh only if the surface is part of the boundary of a volume; the line elements discretizing a curve will be forced to be edges of tetrahedra in the final 3D mesh only if the curve is part of the boundary of a surface, itself part of the boundary of a volume; a single node discretizing a point in the middle of a volume will be forced to be a vertex of one of the tetrahedra in the final 3D mesh only if this point is connected to a curve, itself part of the boundary of a surface, itself part of the boundary of a volume. This automatically assures the conformity of the mesh when, for example, two surfaces share a common line. But this also implies that the discretization of an "isolated" (n-1)-th dimensional entity inside an n-th dimensional entity does *not* constrain the n-th dimensional mesh—unless it is explicitly told to do so (see Section 6.3.3 [Miscellaneous mesh commands], page 57). Every meshing step is constrained by a "size field" (sometimes called "characteristic length field"), which prescribes the desired size of the elements in the mesh. This size field can be uniform, specified by values associated with points in the geometry, or defined by general “fields” (for example related to the distance to some boundary, to a arbitrary scalar field defined on another mesh, etc.).

For each meshing step, all structured mesh directives are executed first, and serve as additional constraints for the unstructured parts[1].

1.3 Solver: external solver interface

External solvers can be interfaced with Gmsh through Unix or TCP/IP sockets, which permits to modify solver parameters, launch external computations and process the results directly from within Gmsh’s post-processing module. The default solver interfaced with Gmsh is GetDP (`http://geuz.org/getdp/`). Examples on how to interface other solvers are available in the source distribution (in the ‘`utils/solvers/`’ directory).

1.4 Post-processing: scalar, vector and tensor field visualization

Gmsh can load and manipulate multiple post-processing scalar, vector or tensor maps along with the geometry and the mesh. Scalar fields are represented by iso-value lines/surfaces or color maps, while vector fields are represented by three-dimensional arrows or displacement maps. Post-processing functions include section computation, offset, elevation, boundary and component extraction, color map and range modification, animation, vector graphic output, etc. All the post-processing options can be accessed either interactively or through the input script files. Scripting permits to automate all post-processing operations, as for example to create animations. User-defined operations can also be performed on post-processing views through dynamically loadable plugins.

1.5 What Gmsh is pretty good at ...

Here is a tentative list of what Gmsh does best:

- quickly describe simple and/or “repetitive” geometries, thanks to user-defined functions, loops, conditionals and includes (see Section 4.5 [User-defined functions], page 24, Section 4.6 [Loops and conditionals], page 25, and Section 4.7 [General commands], page 25);
- parametrize these geometries. Gmsh’s scripting language enables all commands and command arguments to depend on previous calculations (see Section 4.2 [Expressions], page 19, and Section 5.1 [Geometry commands], page 31);
- generate 1D, 2D and 3D simplicial (i.e., using line segments, triangles and tetrahedra) finite element meshes for CAD models in their native format (without translations) when linked with the appropriate CAD kernel (see Chapter 6 [Mesh module], page 39);
- specify target element sizes accurately. Gmsh provides several mechanisms to control the size of the elements in the final mesh: through interpolation from sizes specified at

[1] Note that mixing structured volume grids with unstructured volume grids generated with the default 3D Delaunay algorithm can result, in certain cases, to non-conform surface meshes on their shared boundary. If this happens, you may consider using the frontal algorithm for the unstructured part.

geometry points or using flexible mesh size fields (see Section 6.3 [Mesh commands], page 41);

- create simple extruded geometries and meshes (see Section 5.1 [Geometry commands], page 31, and Section 6.3 [Mesh commands], page 41);
- interact with external solvers through a simple client-server architecture (see Chapter 7 [Solver module], page 59);
- visualize and export computational results in a great variety of ways. Gmsh can display scalar, vector and tensor datasets, perform various operations on the resulting post-processing views (see Chapter 8 [Post-processing module], page 61), can export plots in many different formats (see Section B.1 [General options list], page 131), and can generate complex animations (see Chapter 4 [General tools], page 19, and Section A.8 [t8.geo], page 118);
- run on low end machines and/or machines with no graphical interface. Gmsh can be compiled with or without the GUI, and all versions can be used either interactively or directly from the command line (see Chapter 3 [Running Gmsh on your system], page 11);
- configure your preferred options. Gmsh has a large number of configuration options that can be set interactively using the GUI, scattered inside command files, changed on the fly in scripts, set in per-user configuration files, or specified on the command-line (see Chapter 3 [Running Gmsh on your system], page 11 and Appendix B [Options], page 131);
- and do all the above on various platforms (Windows, Mac and Unix), for free (see [Copying conditions], page 3), using simple script files and/or a small but powerful GUI.

1.6 . . . and what Gmsh is not so good at

As of version 2.7, here are some known weaknesses of Gmsh:

- the BRep approach for describing geometries can become inconvenient/inefficient for large models. For complex models, or if you want to use a solid-modeler approach, you should link Gmsh with an external CAD kernel and import native files directly. (The binary versions available on `http://geuz.org/gmsh/` are linked with the free CAD kernel OpenCASCADE, which enables native BREP import as well as STEP and IGES import.)
- Gmsh's internal CAD engine is fairly limited: it only handles simple primitives and does not perform any complex geometrical operations (e.g. no calculation of intersections). For such features you should link Gmsh with an external CAD kernel (see above).
- Gmsh is not a multi-bloc mesh generator: all meshes produced by Gmsh are conforming in the sense of finite element meshes;
- Gmsh's user interface is only exposing a limited number of the available features, and many aspects of the interface could be enhanced (especially manipulators).
- Gmsh's scripting language is fairly limited, providing only very crude loop controls and user-defined functions, with no local variables.
- there is no global "undo" capability. You will often need to edit a text file to correct mistakes.

If you have the skills and some free time, feel free to join the project: we gladly accept any code contributions (see Appendix C [Information for developers], page 199) to remedy the aforementioned (and all other) shortcomings!

1.7 Bug reports

If you think you have found a bug in Gmsh, you can report it by email to the public Gmsh mailing list at `gmsh@geuz.org`, or file it directly into our bug tracking database at `https://geuz.org/trac/gmsh/report` (login: gmsh, password: gmsh). Please send as precise a description of the problem as you can, including sample input files that produce the bug. Don't forget to mention both the version of Gmsh and the version of your operation system (see Section 3.3 [Command-line options], page 12 to see how to get this information).

See Appendix D [Frequently asked questions], page 201, and the bug tracking system to see which problems we already know about.

2 How to read this reference manual?

Gmsh can be used at three levels:

1. as a stand-alone graphical program, driven by an interactive graphical user interface (GUI);
2. as a stand-alone script-driven program;
3. as a library.

You can skip most of this reference manual if you only want to use Gmsh at the first level (i.e., interactively with the GUI). Just read the next chapter (see Chapter 3 [Running Gmsh on your system], page 11) to learn how to launch Gmsh on your system, then go experiment with the GUI and the tutorial files (see Appendix A [Tutorial], page 105) provided in the distribution. Screencasts that show how to use the GUI are available here: `http://geuz.org/gmsh/screencasts/`.

The aim of the reference manual is to explain everything you need to use Gmsh at the second level, i.e., using the built-in scripting language. A Gmsh script file is an ASCII text file that contains instructions in Gmsh's built-in scripting language. Such a file is interpreted by Gmsh's parser, and can be given any extension (or no extension at all). By convention, Gmsh uses the '`.geo`' extension for geometry scripts, and the '`.pos`' extension for parsed post-processing datasets. Once you master the tutorial (read the source files: they are heavily commented!), start reading chapter Chapter 4 [General tools], page 19, then proceed with the next four chapters, which detail the syntax of the geometry, mesh, solver and post-processing scripting commands. You will see that most of the interactive actions in the GUI have a direct equivalent in the scripting language. If you want to use Gmsh as a pre- or post-processor for your own software, you will also want to learn about the non-scripting input/output files that Gmsh can read/write. In addition to Gmsh's native "MSH" file format (see Chapter 9 [File formats], page 87), Gmsh can read/write many standard mesh files, depending on how it was built: check the 'File->Save As' menu for a list of available formats.

Finally, to use Gmsh at the third level (i.e., to link the Gmsh library with your own code), you will need to learn the internal Gmsh Application Programming Interface (API). No complete documentation of this API is available yet; a good starting point is Section C.2 [Source code structure], page 199, which gives a short introduction to Gmsh's internal source code structure. Then have a look e.g. at the examples in the '`utils/api_demos/`' directory in the source code. Due to the many possible configuration options (and associated external dependencies), we currently do not distibute precompiled versions of the Gmsh library. To build the library see the instructions in the top-level '`README.txt`' file in the source distribution.

2.1 Syntactic rules used in the manual

Here are the rules we tried to follow when writing this reference manual. Note that metasyntactic variable definitions stay valid throughout the manual (and not only in the sections where the definitions appear).

1. Keywords and literal symbols are printed like `this`.
2. Metasyntactic variables (i.e., text bits that are not part of the syntax, but stand for other text bits) are printed like *this*.

3. A colon (`:`) after a metasyntactic variable separates the variable from its definition.
4. Optional rules are enclosed in `< >` pairs.
5. Multiple choices are separated by `|`.
6. Three dots (. . .) indicate a possible (multiple) repetition of the preceding rule.

3 Running Gmsh on your system

3.1 Interactive mode

To launch Gmsh in interactive mode, just double-click on the Gmsh icon, or type

```
> gmsh
```

at your shell prompt in a terminal. This will open the main Gmsh window, with a tree-like menu on the left, a graphic area on the right, and a status bar at the bottom. (You can detach the tree menu using 'Window->Attach/Detach Menu'.)

To open the first tutorial file (see Appendix A [Tutorial], page 105), select the 'File->Open' menu, and choose '`t1.geo`'. When using a terminal, you can specify the file name directly on the command line, i.e.:

```
> gmsh t1.geo
```

To perform the mesh generation, go to the mesh module (by selecting 'Mesh' in the tree) and choose the dimension ('1D' will mesh all the lines; '2D' will mesh all the surfaces—as well as all the lines if '1D' was not called before; '3D' will mesh all the volumes—and all the surfaces if '2D' was not called before). To save the resulting mesh in the current mesh format click on 'Save', or select the appropriate format and file name with the 'File->Save As' menu. The default mesh file name is based on the name of the current active model, with an appended extension depending on the mesh format[1].

To create a new geometry or to modify an existing geometry, select 'Geometry' in the tree. For example, to create a spline, select 'Elementary', 'Add', 'New' and 'Spline'. You will then be asked to select a list of points, and to type `e` to finish the selection (or `q` to abort it). Once the interactive command is completed, a text string is automatically added at the end of the current script file. You can edit the script file by hand at any time by pressing the 'Edit' button in the 'Geometry' menu and then reloading the model by pressing 'Reload'. For example, it is often faster to define variables and points directly in the script file, and then use the GUI to define the lines, the surfaces and the volumes interactively.

Several files can be loaded simultaneously in Gmsh. When specified on the command line, the first one defines the active model and the others are 'merged' into this model. You can merge such files with the 'File->Merge' menu. For example, to merge the post-processing views contained in the files '`view1.pos`' and '`view5.msh`' together with the geometry of the first tutorial '`t1.geo`', you can type the following command:

```
> gmsh t1.geo view1.pos view5.msh
```

In the Post-Processing module (select 'Post-Processing' in the tree), three items will appear, respectively labeled 'A scalar map', 'Nodal scalar map' and 'Element 1 vector'. In this example the views contain several time steps: you can loop through them with the small "remote-control" icons in the status bar. A mouse click on the view name will toggle the visibility of the selected view, while a click on the arrow button on the right will provide access to the view's options.

[1] Nearly all the interactive commands have keyboard shortcuts: see Section 3.5 [Keyboard shortcuts], page 16, or select 'Help->Keyboard and Mouse Usage' in the menu. For example, to quickly save a mesh, you can press *`Ctrl+Shift+s`*.

Note that all the options specified interactively can also be directly specified in the script files. You can save the current options of the current active model with the 'File->Save Model Options'. This will create a new option file with the same filename as the active model, but with an extra '`.opt`' extension added. The next time you open this model, the associated options will be automatically loaded, too. To save the current options as your default preferences for all future Gmsh sessions, use the 'File->Save Options As Default' menu instead. Finally, you can also save the current options in an arbitrary file by choosing the 'Gmsh options' format in 'File->Save As'.

For more information about available options (and how to reset them to their default values), see Appendix B [Options], page 131. A full list of options with their current values is also available in the 'Help->Current Options' menu.

3.2 Non-interactive mode

Gmsh can be run non-interactively in 'batch' mode, without GUI[2]. For example, to mesh the first tutorial in batch mode, just type:

```
> gmsh t1.geo -2
```

To mesh the same example, but with the background mesh available in the file '`bgmesh.pos`', type:

```
> gmsh t1.geo -2 -bgm bgmesh.pos
```

For the list of all command-line options, see Section 3.3 [Command-line options], page 12.

3.3 Command-line options

Geometry options:

`-0` Output unrolled geometry, then exit

`-tol float`
Set geometrical tolerance

`-match` Match geometries and meshes

Mesh options:

`-1, -2, -3`
Perform 1D, 2D or 3D mesh generation, then exit

`-format string`
Select output mesh format (auto (default), msh, msh1, msh2, unv, vrml, ply2, stl, mesh, bdf, cgns, p3d, diff, med, ...)

`-vmsh float`
Select msh file version

`-refine` Perform uniform mesh refinement, then exit

`-part int` Partition after batch mesh generation

`-partWeight tri|quad|tet|prism|hex int`
Weight of a triangle/quad/etc. during partitioning

2 If you compile Gmsh without the GUI, this is the only mode you have access to.

`-renumber`
: Renumber the mesh elements after batch mesh generation

`-saveall`
: Save all elements (discard physical group definitions)

`-o file`
: Specify output file name

`-bin`
: Use binary format when available

`-parametric`
: Save vertices with their parametric coordinates

`-numsubedges`
: Set num of subdivisions for high order element display

`-algo string`
: Select mesh algorithm (meshadapt, del2d, front2d, delquad, del3d, front3d, mmg3d)

`-smooth int`
: Set number of mesh smoothing steps

`-order int`
: Set mesh order (1, ..., 5)

`-hoOptimize`
: Optimize high order meshes

`-hoMindisto float`
: Min high-order element quality before optim (0.0->1.0)

`-hoNLayers int`
: Number of high order element layers to optimize

`-hoElasticity float`
: Poisson ration for elasticity analogy (nu in [-1.0,0.5])

`-optimize[_netgen]`
: Optimize quality of tetrahedral elements

`-optimize_lloyd`
: Optimize 2D meshes using Lloyd algorithm

`-microstructure`
: Generate polycrystal Voronoi geometry

`-clscale float`
: Set global mesh element size scaling factor

`-clmin float`
: Set minimum mesh element size

`-clmax float`
: Set maximum mesh element size

`-anisoMax float`
: Set maximum anisotropy (only used in bamg for now)

`-smoothRatio float`
: Set smoothing ration between mesh sizes at nodes of a same edge (only used in bamg)

`-clcurv`
: Automatically compute element sizes from curvatures

`-epslc1d`
: Set accuracy of evaluation of LCFIELD for 1D mesh

`-swapangle`
: Set the threshold angle (in degree) between two adjacent faces below which a swap is allowed

`-rand float`
: Set random perturbation factor

`-bgm file`
: Load background mesh from file

`-check`
: Perform various consistency checks on mesh

`-mpass int`
: Do several passes on the mesh for complex backround fields

`-ignorePartBound`
: Ignore partitions boundaries

Post-processing options:

`-link int`
: Select link mode between views (0, 1, 2, 3, 4)

`-combine`
: Combine views having identical names into multi-time-step views

Display options:

`-n`
: Hide all meshes and post-processing views on startup

`-nodb`
: Disable double buffering

`-fontsize int`
: Specify the font size for the GUI

`-theme string`
: Specify FLTK GUI theme

`-display string`
: Specify display

`-camera`
: Use camera mode view;

`-stereo`
: OpenGL quad-buffered stereo rendering (requires special graphic card)

Other options:

`-`
: Parse input files, then exit

`-a, -g, -m, -s, -p`
: Start in automatic, geometry, mesh, solver or post-processing mode

`-pid`
: Print process id on stdout

`-listen`
: Always listen to incoming connection requests

`-watch pattern`
: Pattern of files to merge as they become available

`-v int`
: Set verbosity level

`-nopopup`
: Don't popup dialog windows in scripts

`-string "string"`
: Parse option string at startup

`-option file`
: Parse option file at startup

`-convert files`
: Convert files into latest binary formats, then exit

`-version`
: Show version number

`-info`
: Show detailed version information

`-help`
: Show command line usage

3.4 Mouse actions

`Move`
: - Highlight the entity under the mouse pointer and display its properties
: - Resize a lasso zoom or a lasso (un)selection

`Left button`
: - Rotate
: - Select an entity
: - Accept a lasso zoom or a lasso selection

`Ctrl+Left button`
: Start a lasso zoom or a lasso (un)selection

`Middle button`
: - Zoom
: - Unselect an entity
: - Accept a lasso zoom or a lasso unselection

`Ctrl+Middle button`
: Orthogonalize display

`Right button`
: - Pan
: - Cancel a lasso zoom or a lasso (un)selection
: - Pop-up menu on post-processing view button

`Ctrl+Right button`
: Reset to default viewpoint

For a 2 button mouse, Middle button = Shift+Left button.

For a 1 button mouse, Middle button = Shift+Left button, Right button = Alt+Left button.

3.5 Keyboard shortcuts

(On Mac Ctrl is replaced by Cmd (the 'Apple key') in the shortcuts below.)

Left arrow	Go to previous time step
Right arrow	Go to next time step
Up arrow	Make previous view visible
Down arrow	Make next view visible
0	Reload project file
1 or F1	Mesh lines
2 or F2	Mesh surfaces
3 or F3	Mesh volumes
Escape	Cancel lasso zoom/selection, toggle mouse selection ON/OFF
g	Go to geometry module
m	Go to mesh module
p	Go to post-processing module
s	Go to solver module
Shift+a	Bring all windows to front
Shift+g	Show geometry options
Shift+m	Show mesh options
Shift+o	Show general options
Shift+p	Show post-processing options
Shift+s	Show solver options
Shift+u	Show post-processing view plugins
Shift+w	Show post-processing view options
Shift+Escape	Enable full mouse selection
Ctrl+i	Show statistics window
Ctrl+d	Attach/detach menu
Ctrl+f	Enter full screen
Ctrl+l	Show message console
Ctrl+m	Minimize window
Ctrl+n	Create new project file

Ctrl+o	Open project file
Ctrl+q	Quit
Ctrl+r	Rename project file
Ctrl+s	Save file as
Shift+Ctrl+c	Show clipping plane window
Shift+Ctrl+m	Show manipulator window
Shift+Ctrl+n	Show option window
Shift+Ctrl+o	Merge file(s)
Shift+Ctrl+s	Save mesh in default format
Shift+Ctrl+u	Show plugin window
Shift+Ctrl+v	Show visibility window
Alt+a	Loop through axes modes
Alt+b	Hide/show bounding boxes
Alt+c	Loop through predefined color schemes
Alt+e	Hide/Show element outlines for visible post-pro views
Alt+f	Change redraw mode (fast/full)
Alt+h	Hide/show all post-processing views
Alt+i	Hide/show all post-processing view scales
Alt+l	Hide/show geometry lines
Alt+m	Toggle visibility of all mesh entities
Alt+n	Hide/show all post-processing view annotations
Alt+o	Change projection mode (orthographic/perspective)
Alt+p	Hide/show geometry points
Alt+r	Loop through range modes for visible post-pro views
Alt+s	Hide/show geometry surfaces
Alt+t	Loop through interval modes for visible post-pro views
Alt+v	Hide/show geometry volumes
Alt+w	Enable/disable all lighting

Alt+x	Set X view
Alt+y	Set Y view
Alt+z	Set Z view
Alt+Shift+a	Hide/show small axes
Alt+Shift+b	Hide/show mesh volume faces
Alt+Shift+d	Hide/show mesh surface faces
Alt+Shift+l	Hide/show mesh lines
Alt+Shift+o	Adjust projection parameters
Alt+Shift+p	Hide/show mesh points
Alt+Shift+s	Hide/show mesh surface edges
Alt+Shift+v	Hide/show mesh volume edges
Alt+Shift+w	Reverse all mesh normals
Alt+Shift+x	Set -X view
Alt+Shift+y	Set -Y view
Alt+Shift+z	Set -Z view

4 General tools

This chapter describes the general commands and options that can be used in Gmsh's script files. By "general", we mean "not specifically related to one of the geometry, mesh, solver or post-processing modules". Commands peculiar to these modules will be introduced in Chapter 5 [Geometry module], page 31, Chapter 6 [Mesh module], page 39, Chapter 7 [Solver module], page 59, and Chapter 8 [Post-processing module], page 61, respectively.

4.1 Comments

Gmsh script files support both C and C++ style comments:

1. any text comprised between `/*` and `*/` pairs is ignored;
2. the rest of a line after a double slash `//` is ignored.

These commands won't have the described effects inside double quotes or inside keywords. Also note that 'white space' (spaces, tabs, new line characters) is ignored inside all expressions.

4.2 Expressions

The two constant types used in Gmsh scripts are *real* and *string* (there is no integer type). These types have the same meaning and syntax as in the C or C++ programming languages.

4.2.1 Floating point expressions

Floating point expressions (or, more simply, "expressions") are denoted by the metasyntactic variable *expression* (remember the definition of the syntactic rules in Section 2.1 [Syntactic rules], page 9), and are evaluated during the parsing of the script file:

```
expression:
  real |
  string |
  string [ expression ] |
  # string [ ] |
  ( expression ) |
  operator-unary-left expression |
  expression operator-unary-right |
  expression operator-binary expression |
  expression operator-ternary-left expression operator-ternary-right ex-
pression |
  built-in-function |
  real-option |
  StrFind(char-expression, char-expression) |
  GetValue("string", expression)
```

Such *expressions* are used in most of Gmsh's scripting commands. The third and fourth cases in this definition permit to extract one item from a list (see below) and get the size of a list, respectively. The operators *operator-unary-left*, *operator-unary-right*, *operator-binary*, *operator-ternary-left* and *operator-ternary-right* are defined in Section 4.3 [Operators], page 21. For the definition of *built-in-function*s, see Section 4.4 [Built-in functions], page 23. The various *real-option*s are listed in Appendix B [Options], page 131. `StrFind` searches the first *char-expression* for any occurrence of the second *char-expression*.

The last case in the definition allows to ask the user for a value interactively. For example, inserting `GetValue("Value of parameter alpha?", 5.76)` in an input file will query the user for the value of a certain parameter alpha, assuming the default value is 5.76. If the option `General.NoPopup` is set (see Section B.1 [General options list], page 131), no question is asked and the default value is automatically used.

List of expressions are also widely used, and are defined as:

```
expression-list:
  expression-list-item <, expression-list-item> ...
```

with

```
expression-list-item:
  expression |
  expression : expression |
  expression : expression : expression |
  string [ ] |
  string ( ) |
  List [ string ] |
  string [ { expression-list } ] |
  string ( { expression-list } ) |
  Point { expression } |
  transform |
  extrude
  Point { expression } |
  Point|Line|Surface|Volume "*" |
  Physical Point|Line|Surface|Volume { expression-list }
```

The second case in this last definition permits to create a list containing the range of numbers comprised between two *expression*s, with a unit incrementation step. The third case also permits to create a list containing the range of numbers comprised between two *expression*s, but with a positive or negative incrementation step equal to the third *expression*. The fourth, fifth and sixth cases permit to reference an expression list. The seventh and eight cases permit to reference an expression sublist (whose elements are those corresponding to the indices provided by the *expression-list*). The next two cases permit to retrieve the indices of entities created through geometrical transformations and extrusions (see Section 5.1.6 [Transformations], page 36, and Section 5.1.5 [Extrusions], page 35). The last three cases permit to retrieve the coordinates of a given geometry point (see Section 5.1.1 [Points], page 31), to retrieve the id numbers of all points, lines, surfaces or volumes in the model, or to retrieve the elementary entities making up physical groups.

To see the practical use of such expressions, have a look at the first couple of examples in Appendix A [Tutorial], page 105. Note that, in order to lighten the syntax, you can omit the braces `{}` enclosing an *expression-list* if this *expression-list* only contains a single item. Also note that a braced *expression-list* can be preceded by a minus sign in order to change the sign of all the *expression-list-item*s.

4.2.2 Character expressions

Character expressions are defined as:

```
char-expression:
  "string" |
  Today |
  StrPrefix ( char-expression ) |
  StrRelative ( char-expression ) |
  StrCat ( char-expression , char-expression ) |
  Sprintf ( char-expression , expression-list ) |
  Sprintf ( char-expression ) |
  Sprintf ( char-option ) |
  GetEnv ( char-expression ) |
  GetString ( char-expression , char-expression )
```

The third and fourth cases in this definition permit to take the prefix (e.g. to remove the extension) or the relative path of a string. The fifth case permits to concatenate two character expressions, and the sixth and seventh are equivalent to the `sprintf` C function (where *char-expression* is a format string that can contain floating point formatting characters: `%e`, `%g`, etc.). The eigth case permits to use the value of a *char-option* as a *char-expression*. The ninth case gets the value of an environment variable from the operating system. The last case in the definition allows to ask the user for a value interactively. The various *char-options* are listed in Appendix B [Options], page 131.

Character expressions are mostly used to specify non-numeric options and input/output file names. See Section A.8 [t8.geo], page 118, for an interesting usage of *char-expressions* in an animation script.

4.2.3 Color expressions

Colors expressions are hybrids between fixed-length braced *expression-lists* and *strings*:

```
color-expression:
  string |
  { expression, expression, expression } |
  { expression, expression, expression, expression } |
  color-option
```

The first case permits to use the X Windows names to refer to colors, e.g., `Red`, `SpringGreen`, `LavenderBlush3`, ... (see '`Common/Colors.h`' in the source code for a complete list). The second case permits to define colors by using three expressions to specify their red, green and blue components (with values comprised between 0 and 255). The third case permits to define colors by using their red, green and blue color components as well as their alpha channel. The last case permits to use the value of a *color-option* as a *color-expression*. The various *color-options* are listed in Appendix B [Options], page 131.

See Section A.3 [t3.geo], page 109, for an example of the use of color expressions.

4.3 Operators

Gmsh's operators are similar to the corresponding operators in C and C++. Here is the list of the unary, binary and ternary operators currently implemented.

operator-unary-left:

`-`	Unary minus.
`!`	Logical not.

operator-unary-right:

`++`	Post-incrementation.
`--`	Post-decrementation.

operator-binary:

`^`	Exponentiation.
`*`	Multiplication.
`/`	Division.
`%`	Modulo.
`+`	Addition.
`-`	Subtraction.
`==`	Equality.
`!=`	Inequality.
`>`	Greater.
`>=`	Greater or equality.
`<`	Less.
`<=`	Less or equality.
`&&`	Logical 'and'.
`||`	Logical 'or'. (Warning: the logical 'or' always implies the evaluation of both arguments. That is, unlike in C or C++, the second operand of `||` is evaluated even if the first one is true).

operator-ternary-left:

`?`

operator-ternary-right:

`:`	The only ternary operator, formed by *operator-ternary-left* and *operator-ternary-right*, returns the value of its second argument if the first argument is non-zero; otherwise it returns the value of its third argument.

The evaluation priorities are summarized below[1] (from stronger to weaker, i.e., `*` has a highest evaluation priority than `+`). Parentheses `()` may be used anywhere to change the order of evaluation:

1. `()`, `[]`, `.`, `#`
2. `^`
3. `!`, `++`, `--`, `-` (unary)

[1] The affectation operators are introduced in Section 4.7 [General commands], page 25.

4. `*`, `/`, `%`
5. `+`, `-`
6. `<`, `>`, `<=`, `>=`
7. `==`, `!=`
8. `&&`
9. `||`
10. `?:`
11. `=`, `+=`, `-=`, `*=`, `/=`

4.4 Built-in functions

A built-in function is composed of an identifier followed by a pair of parentheses containing an *expression-list* (the list of its arguments)[2]. Here is the list of the built-in functions currently implemented:

build-in-function:

`Acos (` ***`expression`*** `)`
: Arc cosine (inverse cosine) of an *expression* in [-1,1]. Returns a value in [0,Pi].

`Asin (` ***`expression`*** `)`
: Arc sine (inverse sine) of an *expression* in [-1,1]. Returns a value in [-Pi/2,Pi/2].

`Atan (` ***`expression`*** `)`
: Arc tangent (inverse tangent) of *expression*. Returns a value in [-Pi/2,Pi/2].

`Atan2 (` ***`expression`***`,` ***`expression`*** `)`
: Arc tangent (inverse tangent) of the first *expression* divided by the second. Returns a value in [-Pi,Pi].

`Ceil (` ***`expression`*** `)`
: Rounds *expression* up to the nearest integer.

`Cos (` ***`expression`*** `)`
: Cosine of *expression*.

`Cosh (` ***`expression`*** `)`
: Hyperbolic cosine of *expression*.

`Exp (` ***`expression`*** `)`
: Returns the value of e (the base of natural logarithms) raised to the power of *expression*.

`Fabs (` ***`expression`*** `)`
: Absolute value of *expression*.

`Fmod (` ***`expression`***`,` ***`expression`*** `)`
: Remainder of the division of the first *expression* by the second, with the sign of the first.

[2] For compatibility with GetDP (`http://geuz.org/getdp/`), parentheses can be replaced by brackets `[]`.

`Floor (` *`expression`* `)`
: Rounds *expression* down to the nearest integer.

`Hypot (` *`expression`*`,` *`expression`* `)`
: Returns the square root of the sum of the square of its two arguments.

`Log (` *`expression`* `)`
: Natural logarithm of *expression* (*expression* `>` 0).

`Log10 (` *`expression`* `)`
: Base 10 logarithm of *expression* (*expression* `>` 0).

`Modulo (` *`expression`*`,` *`expression`* `)`
: see `Fmod(` *`expression`*`,` *`expression`* `)`.

`Rand (` *`expression`* `)`
: Random number between zero and *expression*.

`Sqrt (` *`expression`* `)`
: Square root of *expression* (*expression* `>=` 0).

`Sin (` *`expression`* `)`
: Sine of *expression*.

`Sinh (` *`expression`* `)`
: Hyperbolic sine of *expression*.

`Tan (` *`expression`* `)`
: Tangent of *expression*.

`Tanh (` *`expression`* `)`
: Hyperbolic tangent of *expression*.

4.5 User-defined functions

User-defined functions take no arguments, and are evaluated as if a file containing the function body was included at the location of the `Call` statement.

`Function` *`string`*
: Begins the declaration of a user-defined function named *string*. The body of the function starts on the line after '`Function` *`string`*', and can contain any Gmsh command.

`Return`
: Ends the body of the current user-defined function. Function declarations cannot be imbricated.

`Call` *`string`*`;`
: Executes the body of a (previously defined) function named *string*.

See Section A.5 [t5.geo], page 113, for an example of a user-defined function. A shortcoming of Gmsh's scripting language is that all variables are "public". Variables defined inside the body of a function will thus be available outside, too!

4.6 Loops and conditionals

Loops and conditionals are defined as follows, and can be imbricated:

`For ( `*`expression`*` : `*`expression`*` )`
: Iterates from the value of the first *expression* to the value of the second *expression*, with a unit incrementation step. At each iteration, the commands comprised between '`For ( `*`expression`*` : `*`expression`*` )`' and the matching `EndFor` are executed.

`For ( `*`expression`*` : `*`expression`*` : `*`expression`*` )`
: Iterates from the value of the first *expression* to the value of the second *expression*, with a positive or negative incrementation step equal to the third *expression*. At each iteration, the commands comprised between '`For ( `*`expression`*` : `*`expression`*` : `*`expression`*` )`' and the matching `EndFor` are executed.

`For `*`string`*` In { `*`expression`*` : `*`expression`*` }`
: Iterates from the value of the first *expression* to the value of the second *expression*, with a unit incrementation step. At each iteration, the value of the iterate is affected to an expression named *string*, and the commands comprised between '`For `*`string`*` In { `*`expression`*` : `*`expression`*` }`' and the matching `EndFor` are executed.

`For `*`string`*` In { `*`expression`*` : `*`expression`*` : `*`expression`*` }`
: Iterates from the value of the first *expression* to the value of the second *expression*, with a positive or negative incrementation step equal to the third *expression*. At each iteration, the value of the iterate is affected to an expression named *string*, and the commands comprised between '`For `*`string`*` In { `*`expression`*` : `*`expression`*` : `*`expression`*` }`' and the matching `EndFor` are executed.

`EndFor`
: Ends a matching `For` command.

`If ( `*`expression`*` )`
: The body enclosed between '`If ( `*`expression`*` )`' and the matching `Endif` is evaluated if *expression* is non-zero.

`EndIf`
: Ends a matching `If` command.

See Section A.5 [t5.geo], page 113, for an example of `For` and `If` commands. Gmsh does not provide any `Else` (or similar) command at the time of this writing.

4.7 General commands

The following commands can be used anywhere in a Gmsh script:

`string`` = `*`expression`*`;`
: Creates a new expression identifier *string*, or affects *expression* to an existing expression identifier. Thirteen expression identifiers are predefined (hardcoded in Gmsh's parser):

  `Pi`
  : Returns 3.1415926535897932.

  `GMSH_MAJOR_VERSION`
  : Returns Gmsh's major version number.

`GMSH_MINOR_VERSION`
: Returns Gmsh's minor version number.

`GMSH_PATCH_VERSION`
: Returns Gmsh's patch version number.

`MPI_Size`
: Returns the number of processors on which Gmsh is running (always 1, except if you compiled Gmsh's parallel extensions).

`MPI_Rank`
: Returns the rank of the current processor.

`newp`
: Returns the next available point number. As explained in Chapter 5 [Geometry module], page 31, a unique number must be associated with every geometrical point: `newp` permits to know the highest number already attributed (plus one). This is mostly useful when writing user-defined functions (see Section 4.5 [User-defined functions], page 24) or general geometric primitives, when one does not know *a priori* which numbers are already attributed, and which ones are still available.

`newl`
: Returns the next available line number.

`news`
: Returns the next available surface number.

`newv`
: Returns the next available volume number.

`newll`
: Returns the next available line loop number.

`newsl`
: Returns the next available surface loop number.

`newreg`
: Returns the next available region number. That is, `newreg` returns the maximum of `newp`, `newl`, `news`, `newv`, `newll`, `newsl` and all physical entity numbers[3].

`string` `= { };`
: Creates a new expression list identifier ***`string`*** with an empty list.

`string` `= {` ***`expression-list`*** `};`
: Creates a new expression list identifier ***`string`*** with the list *expression-list*, or affects *expression-list* to an existing expression list identifier. The following syntax is also allowed: ***`string`***`[] = {` ***`expression-list`*** `};`

`string` `[ {` ***`expression-list`*** `} ] = {` ***`expression-list`*** `};`
: Affects each item in the right hand side *expression-list* to the elements (indexed by the left hand side *expression-list*) of an existing expression list identifier. The two *expression-lists* must contain the same number of items.

`string` `( {` ***`expression-list`*** `} ) = {` ***`expression-list`*** `};`
: Same as above.

`string` `+=` ***`expression`***`;`
: Adds and affects *expression* to an existing expression identifier.

[3] For compatibility purposes, the behavior of `newl`, `news`, `newv` and `newreg` can be modified with the `Geometry.OldNewReg` option (see Section B.2 [Geometry options list], page 156).

`string -= expression ;`
: Subtracts and affects *expression* to an existing expression identifier.

`string *= expression ;`
: Multiplies and affects *expression* to an existing expression identifier.

`string /= expression ;`
: Divides and affects *expression* to an existing expression identifier.

`string += { expression-list };`
: Appends *expression-list* to an existing expression list or creates a new expression list with *expression-list*.

`string -= { expression-list };`
: Removes the items in *expression-list* from the existing expression list.

`string [ { expression-list } ] += { expression-list };`
: Adds and affects, item per item, the right hand side *expression-list* to an existing expression list identifier.

`string [ { expression-list } ] -= { expression-list };`
: Subtracts and affects, item per item, the right hand side *expression-list* to an existing expression list identifier.

`string [ { expression-list } ] *= { expression-list };`
: Multiplies and affects, item per item, the right hand side *expression-list* to an existing expression list identifier.

`string [ { expression-list } ] /= { expression-list };`
: Divides and affects, item per item, the right hand side *expression-list* to an existing expression list identifier.

`string = char-expression ;`
: Creates a new character expression identifier `string` with a given *char-expression*.

`real-option = expression ;`
: Affects *expression* to a real option.

`char-option = char-expression ;`
: Affects *char-expression* to a character option.

`color-option = color-expression ;`
: Affects *color-expression* to a color option.

`real-option += expression ;`
: Adds and affects *expression* to a real option.

`real-option -= expression ;`
: Subtracts and affects *expression* to a real option.

`real-option *= expression ;`
: Multiplies and affects *expression* to a real option.

`real-option /= expression ;`
: Divides and affects *expression* to a real option.

`Abort;`
: Aborts the current script.

`Exit;`
: Exits Gmsh.

`Printf (` *`char-expression`* `<,` *`expression-list`*`> );`
: Prints a character expression in the information window and/or on the terminal. `Printf` is equivalent to the `printf` C function: *char-expression* is a format string that can contain formatting characters (`%f`, `%e`, etc.). Note that all *expression*s are evaluated as floating point values in Gmsh (see Section 4.2 [Expressions], page 19), so that only valid floating point formatting characters make sense in *char-expression*. See Section A.5 [t5.geo], page 113, for an example of the use of `Printf`.

`Printf (` *`char-expression`* `,` *`expression-list`* `) >` *`char-expression`*`;`
: Same as `Printf` above, but output the expression in a file.

`Printf (` *`char-expression`* `,` *`expression-list`* `) >>` *`char-expression`*`;`
: Same as `Printf` above, but appends the expression at the end of the file.

`Error (` *`char-expression`* `<,` *`expression-list`*`> );`
: Same as `Printf`, but raises an error.

`Merge` *`char-expression`*`;`
: Merges a file named *char-expression*. This command is equivalent to the 'File->Merge' menu in the GUI. If the path in *char-expression* is not absolute, *char-expression* is appended to the path of the current file.

`Draw;`
: Redraws the scene.

`BoundingBox;`
: Recomputes the bounding box of the scene (which is normally computed only after new geometrical entities are added or after files are included or merged). The bounding box is computed as follows:

1. If there is a mesh (i.e., at least one mesh vertex), the bounding box is taken as the box enclosing all the mesh vertices;
2. If there is no mesh but there is a geometry (i.e., at least one geometrical point), the bounding box is taken as the box enclosing all the geometrical points;
3. If there is no mesh and no geometry, but there are some post-processing views, the bounding box is taken as the box enclosing all the primitives in the views.

`BoundingBox {` *`expression`*`,` *`expression`*`,` *`expression`*`,` *`expression`*`,` *`expression`*`,` *`expression`* `};`
: Forces the bounding box of the scene to the given *expression*s (X min, X max, Y min, Y max, Z min, Z max).

`Delete Model;`
: Deletes the current model (all geometrical entities and their associated meshes).

`Delete Physicals;`
: Deletes all physical groups.

`Delete Variables;`
: Deletes all the expressions.

`Delete Options;`
: Deletes the current options and revert to the default values.

`Delete` *`string`*`;`
: Deletes the expression *string*.

`Mesh` *`expression`*`;`
: Generates *expression*-D mesh.

`RefineMesh;`
: Refines the current mesh by splitting all elements. If `Mesh.SecondOrderLinear` is set, the new vertices are inserted by linear interpolatinon. Otherwise they are snapped on the actual geometry.

`Print` *`char-expression`*`;`
: Prints the graphic window in a file named *char-expression*, using the current `Print.Format` (see Section B.1 [General options list], page 131). If the path in *char-expression* is not absolute, *char-expression* is appended to the path of the current file.

`Sleep` *`expression`*`;`
: Suspends the execution of Gmsh during *expression* seconds.

`SystemCall` *`char-expression`*`;`
: Executes a (blocking) system call.

`NonBlockingSystemCall` *`char-expression`*`;`
: Executes a (non-blocking) system call.

`SetName` *`char-expression`*`;`
: Changes the name of the current model.

`SyncModel;`
: Forces an immediate transfer from the old geometrical database into the new one (this transfer normally occurs right after a file is read).

`Include` *`char-expression`*`;`
: Includes the file named *char-expression* at the current position in the input file. The include command should be given on a line of its own. If the path in *char-expression* is not absolute, *char-expression* is appended to the path of the current file.

4.8 General options

The list of all the general *char-option*s, *real-option*s and *color-option*s (in that order—check the default values to see the actual types) is given in Section B.1 [General options list], page 131. Most of these options are accessible in the GUI, but not all of them. When running Gmsh interactively, changing an option in the script file will modify the option in the GUI in real time. This permits for example to resize the graphical window in a script, or to interact with animations in the script and in the GUI at the same time.

5 Geometry module

Gmsh's geometry module provides a simple CAD engine, using a boundary representation ("BRep") approach: you need to first define points (using the `Point` command: see below), then lines (using `Line`, `Circle`, `Spline`, . . . , commands or by extruding points), then surfaces (using for example the `Plane Surface` or `Ruled Surface` commands, or by extruding lines), and finally volumes (using the `Volume` command or by extruding surfaces).

These geometrical entities are called "elementary" in Gmsh's jargon, and are assigned identification numbers when they are created:

1. each elementary point must possess a unique identification number;
2. each elementary line must possess a unique identification number;
3. each elementary surface must possess a unique identification number;
4. each elementary volume must possess a unique identification number.

Elementary geometrical entities can then be manipulated in various ways, for example using the `Translate`, `Rotate`, `Scale` or `Symmetry` commands. They can be deleted with the `Delete` command, provided that no higher-dimension entity references them.

Groups of elementary geometrical entities can also be defined and are called "physical" entities. These physical entities cannot be modified by geometry commands: their only purpose is to assemble elementary entities into larger groups, possibly modifying their orientation, so that they can be referred to by the mesh module as single entities. As is the case with elementary entities, each physical point, physical line, physical surface or physical volume must be assigned a unique identification number. See Chapter 6 [Mesh module], page 39, for more information about how physical entities affect the way meshes are saved.

5.1 Geometry commands

The next subsections describe all the available geometry commands. These commands can be used anywhere in a Gmsh script file. Note that the following general syntax rule is followed for the definition of geometrical entities: "If an *expression* defines a new entity, it is enclosed between parentheses. If an *expression* refers to a previously defined entity, it is enclosed between braces."

5.1.1 Points

`Point ( expression ) = { expression, expression, expression <, expression > };`

Creates an elementary point. The *expression* inside the parentheses is the point's identification number; the three first *expressions* inside the braces on the right hand side give the three X, Y and Z coordinates of the point in the three-dimensional Euclidean space; the optional last *expression* sets the prescribed mesh element size at that point. See Section 6.3.1 [Specifying mesh element sizes], page 41, for more information about how this value is used in the meshing process.

`Physical Point ( expression | char-expression ) = { expression-list };`

Creates a physical point. The *expression* inside the parentheses is the physical point's identification number (if a *char-expression* is given instead, a unique identification number is automatically created); the *expression-list* on the right hand side should contain the identification numbers of all the elementary points that need to be grouped inside the physical point.

5.1.2 Lines

`BSpline ( `*`expression`*` ) = { `*`expression-list`*` };`
: Creates a B-spline curve. The *expression* inside the parentheses is the B-spline curve's identification number; the *expression-list* on the right hand side should contain the identification numbers of all the B-spline's control points. Repeating control points has the expected effect.

`Circle ( `*`expression`*` ) = { `*`expression`*`, `*`expression`*`, `*`expression`*` };`
: Creates a circle arc (strictly) smaller than Pi. The *expression* inside the parentheses is the circle arc's identification number; the first *expression* inside the braces on the right hand side gives the identification number of the start point of the arc; the second *expression* gives the identification number of the center of the circle; the last *expression* gives the identification number of the end point of the arc.

`CatmullRom ( `*`expression`*` ) = { `*`expression-list`*` };`
: `CatmullRom` is a synonym for `Spline`.

`Ellipse ( `*`expression`*` ) = { `*`expression`*`, `*`expression`*`, `*`expression`*`, `*`expression`*` };`
: Creates an ellipse arc. The *expression* inside the parentheses is the ellipse arc's identification number; the first *expression* inside the braces on the right hand side gives the identification number of the start point of the arc; the second *expression* gives the identification number of the center of the ellipse; the third *expression* gives the identification number of any point located on the major axis of the ellipse; the last *expression* gives the identification number of the end point of the arc.

`Line ( `*`expression`*` ) = { `*`expression`*`, `*`expression`*` };`
: Creates a straight line segment. The *expression* inside the parentheses is the line segment's identification number; the two *expression*s inside the braces on the right hand side give identification numbers of the start and end points of the segment.

`Spline ( `*`expression`*` ) = { `*`expression-list`*` };`
: Creates a spline curve. The *expression* inside the parentheses is the spline's identification number; the *expression-list* on the right hand side should contain the identification numbers of all the spline's control points.

`Line Loop ( `*`expression`*` ) = { `*`expression-list`*` };`
: Creates an oriented line loop. The *expression* inside the parentheses is the line loop's identification number; the *expression-list* on the right hand side should contain the identification numbers of all the elementary lines that constitute the line loop. A line loop must be a closed loop, and the elementary lines should be ordered and oriented (using negative identification numbers to specify reverse orientation). If the orientation is correct, but the ordering is wrong, Gmsh will actually reorder the list internally to create a consistent loop. Although Gmsh supports it, it is not recommended to specify multiple line loops (or subloops)

in a single `Line Loop` command. (Line loops are used to create surfaces: see Section 5.1.3 [Surfaces], page 33.)

`Compound Line (` *`expression`* `) = {` *`expression-list`* `};`
: Creates a compound line from several elementary lines. When meshed, a compound line will be reparametrized as a single line, whose mesh can thus cross internal boundaries. The *expression* inside the parentheses is the compound line's identification number; the *expression-list* on the right hand side contains the identification number of the elementary lines that should be reparametrized as a single line. See `Compound Surface` for additional information on compound entities.

`Physical Line (` *`expression`* `|` *`char-expression`* `) = {` *`expression-list`* `};`
: Creates a physical line. The *expression* inside the parentheses is the physical line's identification number (if a *char-expression* is given instead, a unique identification number is automatically created); the *expression-list* on the right hand side should contain the identification numbers of all the elementary lines that need to be grouped inside the physical line. Specifying negative identification numbers in the *expression-list* will reverse the orientation of the mesh elements belonging to the corresponding elementary lines in the saved mesh.

5.1.3 Surfaces

`Plane Surface (` *`expression`* `) = {` *`expression-list`* `};`
: Creates a plane surface. The *expression* inside the parentheses is the plane surface's identification number; the *expression-list* on the right hand side should contain the identification numbers of all the line loops defining the surface. The first line loop defines the exterior boundary of the surface; all other line loops define holes in the surface. A line loop defining a hole should not have any lines in common with the exterior line loop (in which case it is not a hole, and the two surfaces should be defined separately). Likewise, a line loop defining a hole should not have any lines in common with another line loop defining a hole in the same surface (in which case the two line loops should be combined).

`Ruled Surface (` *`expression`* `) = {` *`expression-list`* `} < In Sphere {` *`expression`* `} >;`
: Creates a ruled surface, i.e., a surface that can be interpolated using transfinite interpolation. The *expression* inside the parentheses is the ruled surface's identification number; the first *expression-list* on the right hand side should contain the identification number of a line loop composed of either three or four elementary lines. The optional `In Sphere` argument forces the surface to be a spherical patch (the extra parameter gives the identification number of the center of the sphere).

`Surface Loop (` *`expression`* `) = {` *`expression-list`* `};`
: Creates a surface loop (a shell). The *expression* inside the parentheses is the surface loop's identification number; the *expression-list* on the right hand side should contain the identification numbers of all the elementary surfaces that constitute the surface loop. A surface loop must always represent a closed shell, and the elementary surfaces should be oriented consistently (using negative

identification numbers to specify reverse orientation). (Surface loops are used to create volumes: see Section 5.1.4 [Volumes], page 34.)

`Compound Surface ( expression ) = { expression-list } < Boundary { { expression-list }, { expression-list }, { expression-list }, { expression-list } } > ;`

Creates a compound surface from several elementary surfaces. When meshed, a compound surface will be reparametrized as a single surface, whose mesh can thus cross internal boundaries. Compound surfaces are mostly useful for remeshing discrete models; see "J.-F. Remacle, C. Geuzaine, G. Compere and E. Marchandise, *High Quality Surface Remeshing Using Harmonic Maps*, International Journal for Numerical Methods in Engineering, 2009" for details as well as the wiki for more examples. The *expression* inside the parentheses is the compound surface's identification number; the mandatory *expression-list* on the right hand side contains the identification number of the elementary surfaces that should be reparametrized as a single surface.

`Physical Surface ( expression | char-expression ) = { expression-list };`

Creates a physical surface. The *expression* inside the parentheses is the physical surface's identification number (if a *char-expression* is given instead, a unique identification number is automatically created); the *expression-list* on the right hand side should contain the identification numbers of all the elementary surfaces that need to be grouped inside the physical surface. Specifying negative identification numbers in the *expression-list* will reverse the orientation of the mesh elements belonging to the corresponding elementary surfaces in the saved mesh.

5.1.4 Volumes

`Volume ( expression ) = { expression-list };`

Creates a volume. The *expression* inside the parentheses is the volume's identification number; the *expression-list* on the right hand side should contain the identification numbers of all the surface loops defining the volume. The first surface loop defines the exterior boundary of the volume; all other surface loops define holes in the volume. A surface loop defining a hole should not have any surfaces in common with the exterior surface loop (in which case it is not a hole, and the two volumes should be defined separately). Likewise, a surface loop defining a hole should not have any surfaces in common with another surface loop defining a hole in the same volume (in which case the two surface loops should be combined).

`Compound Volume ( expression ) = { expression-list };`

Creates a compound volume from several elementary volumes. When meshed, a compound volume will be reparametrized as a single volume, whose mesh can thus cross internal boundaries. The *expression* inside the parentheses is the compound volume's identification number; the *expression-list* on the right hand side contains the identification number of the elementary volumes that should be reparametrized as a single volume. See `Compound Surface` for additional information on compound entities.

`Physical Volume ( expression | char-expression ) = { expression-list };`
: Creates a physical volume. The *expression* inside the parentheses is the physical volume's identification number (if a *char-expression* is given instead, a unique identification number is automatically created); the *expression-list* on the right hand side should contain the identification numbers of all the elementary volumes that need to be grouped inside the physical volume.

5.1.5 Extrusions

Lines, surfaces and volumes can also be created through extrusion of points, lines and surfaces, respectively. Here is the syntax of the geometrical extrusion commands (go to Section 6.3.2 [Structured grids], page 55, to see how these commands can be extended in order to also extrude the mesh):

extrude:

`Extrude { expression-list } { extrude-list }`
: Extrudes all elementary entities (points, lines or surfaces) in *extrude-list* using a translation. The *expression-list* should contain three *expressions* giving the X, Y and Z components of the translation vector.

`Extrude { { expression-list }, { expression-list }, expression } { extrude-list }`
: Extrudes all elementary entities (points, lines or surfaces) in *extrude-list* using a rotation. The first *expression-list* should contain three *expressions* giving the X, Y and Z direction of the rotation axis; the second *expression-list* should contain three *expressions* giving the X, Y and Z components of any point on this axis; the last *expression* should contain the rotation angle (in radians).

`Extrude { { expression-list }, { expression-list }, { expression-list }, expression } { extrude-list }`
: Extrudes all elementary entities (points, lines or surfaces) in *extrude-list* using a translation combined with a rotation. The first *expression-list* should contain three *expressions* giving the X, Y and Z components of the translation vector; the second *expression-list* should contain three *expressions* giving the X, Y and Z direction of the rotation axis; the third *expression-list* should contain three *expressions* giving the X, Y and Z components of any point on this axis; the last *expression* should contain the rotation angle (in radians).

with

```
extrude-list:
  Point | Line | Surface { expression-list }; ...
```

As explained in Section 4.2.1 [Floating point expressions], page 19, *extrude* can be used in an expression, in which case it returns a list of identification numbers. By default, the list contains the "top" of the extruded entity at index 0 and the extruded entity at index 1, followed by the "sides" of the extruded entity at indices 2, 3, etc. For example:

```
Point(1) = {0,0,0};
Point(2) = {1,0,0};
Line(1) = {1, 2};
out[] = Extrude{0,1,0}{ Line{1}; };
```

```
Printf("top line = %g", out[0]);
Printf("surface = %g", out[1]);
Printf("side lines = %g and %g", out[2], out[3]);
```

This behaviour can be changed with the `Geometry.ExtrudeReturnLateralEntities` option (see Section B.2 [Geometry options list], page 156).

5.1.6 Transformations

Geometrical transformations can be applied to elementary entities, or to copies of elementary entities (using the `Duplicata` command: see below). The syntax of the transformation commands is:

transform:

`Dilate { {` *`expression-list`* `},` *`expression`* `} {` *`transform-list`* `}`
: Scales all elementary entities in *transform-list* by a factor *expression*. The *expression-list* should contain three *expression*s giving the X, Y and Z direction of the homothetic transformation.

`Rotate { {` *`expression-list`* `}, {` *`expression-list`* `},` *`expression`* `} {` *`transform-list`* `}`
: Rotates all elementary entities in *transform-list* by an angle of *expression* radians. The first *expression-list* should contain three *expression*s giving the X, Y and Z direction of the rotation axis; the second *expression-list* should contain three *expression*s giving the X, Y and Z components of any point on this axis.

`Symmetry {` *`expression-list`* `} {` *`transform-list`* `}`
: Transforms all elementary entities symmetrically to a plane. The *expression-list* should contain four *expression*s giving the coefficients of the plane's equation.

`Translate {` *`expression-list`* `} {` *`transform-list`* `}`
: Translates all elementary entities in *transform-list*. The *expression-list* should contain three *expression*s giving the X, Y and Z components of the translation vector.

`Boundary {` *`transform-list`* `}`
: (Not a transformation per-se.) Returns the boundary of the elementary entities in *transform-list*.

`CombinedBoundary {` *`transform-list`* `}`
: (Not a transformation per-se.) Returns the boundary of the elementary entities, combined as if a single entity, in *transform-list*. Useful to compute the boundary of a complex part.

with

```
transform-list:
  Point | Line | Surface | Volume { expression-list }; ... |
  Duplicata { Point | Line | Surface | Volume { expression-list }; ... } |
  transform
```

5.1.7 Miscellaneous

Here is a list of all other geometry commands currently available:

`Coherence;`
: Removes all duplicate elementary geometrical entities (e.g., points having identical coordinates). Note that Gmsh executes the `Coherence` command automatically after each geometrical transformation, unless `Geometry.AutoCoherence` is set to zero (see Section B.2 [Geometry options list], page 156).

`Delete { Point | Line | Surface | Volume {` *`expression-list`* `}; ... }`
: Deletes all elementary entities whose identification numbers are given in *expression-list*. If an entity is linked to another entity (for example, if a point is used as a control point of a curve), `Delete` has no effect (the line will have to be deleted before the point can).

`Hide { Point | Line | Surface | Volume {` *`expression-list`* `}; ... }`
: Hide the entities listed in *expression-list*, if `General.VisibilityMode` is set to `0` or `1`.

`Hide` *`char-expression`*`;`
: Hide the entity *char-expression*, if `General.VisibilityMode` is set to `0` or `1` (*char-expression* can for example be `"*"`).

`Show { Point | Line | Surface | Volume {` *`expression-list`* `}; ... }`
: Show the entities listed in *expression-list*, if `General.VisibilityMode` is set to `0` or `1`.

`Show` *`char-expression`*`;`
: Show the entity *char-expression*, if `General.VisibilityMode` is set to `0` or `1` (*char-expression* can for example be `"*"`).

5.2 Geometry options

The list of all the options that control the behavior of geometry commands, as well as the way geometrical entities are handled in the GUI, is give in Section B.2 [Geometry options list], page 156.

6 Mesh module

Gmsh's mesh module regroups several 1D, 2D and 3D meshing algorithms, all producing grids conforming in the sense of finite elements (see Section 1.2 [Mesh], page 5):

- The 2D *unstructured* algorithms generate triangles or both triangles and quadrangles (when `Recombine Surface` is used: see Section 6.3.3 [Miscellaneous mesh commands], page 57). The 3D *unstructured* algorithms generate tetrahedra.
- The 2D *structured* algorithms (transfinite and extrusion) generate triangles by default, but quadrangles can be obtained by using the `Recombine` commands (see Section 6.3.2 [Structured grids], page 55, and Section 6.3.3 [Miscellaneous mesh commands], page 57). The 3D *structured* algorithms generate tetrahedra, hexahedra, prisms and pyramids, depending on the type of the surface meshes they are based on.

All meshes can be subdivided to generate fully quadrangular or fully hexahedral meshes with the `Mesh.SubdivisionAlgorihm` option (see Section B.3 [Mesh options list], page 163). However, beware that the quality of subdivided elements initially generated with an *unstructured* algorithm can be quite poor.

6.1 Choosing the right unstructured algorithm

Gmsh currently provides a choice between three 2D unstructured algorithms and between two 3D unstructured algorithms. Each algorithm has its own advantages and disadvantages.

For all 2D unstructured algorithms a Delaunay mesh that contains all the points of the 1D mesh is initially constructed using a divide-and-conquer algorithm[1]. Missing edges are recovered using edge swaps[2]. After this initial step three different algorithms can be applied to generate the final mesh:

1. The "MeshAdapt" algorithm[3] is based on local mesh modifications. This technique makes use of edge swaps, splits, and collapses: long edges are split, short edges are collapsed, and edges are swapped if a better geometrical configuration is obtained.
2. The "Delaunay" algorithm is inspired by the work of the GAMMA team at INRIA[4]. New points are inserted sequentially at the circumcenter of the element that has the largest adimensional circumradius. The mesh is then reconnected using an anisotropic Delaunay criterion.
3. The "Frontal" algorithm is inspired by the work of S. Rebay[5].

These algorithms can be ranked as follows:

---

[1] R. A. Dwyer, *A simple divide-and-conquer algorithm for computing Delaunay triangulations in O(n log n) expected time*, In Proceedings of the second annual symposium on computational geometry, Yorktown Heights, 2–4 June 1986.

[2] N. P. Weatherill, *The integrity of geometrical boundaries in the two-dimensional Delaunay triangulation*, Commun. Appl. Numer. Methods 6(2), pp. 101–109, 1990.

[3] C. 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(11), pp. 1309–1331, 2009.

[4] P.-L. George and P. Frey, *Mesh generation*, Hermes, Lyon, 2000.

[5] S. Rebay, *Efficient unstructured mesh generation by means of Delaunay triangulation and Bowyer-Watson algorithm*, J. Comput. Phys. 106, pp. 25–138, 1993.

	Robustness	Performance	Element quality
MeshAdapt	1	3	2
Delaunay	2	1	2
Frontal	3	2	1

For very complex curved surfaces the "MeshAdapt" algorithm is the best choice. When high element quality is important, the "Frontal" algorithm should be tried. For very large meshes of plane surfaces the "Delaunay" algorithm is the fastest.

The "Automatic" algorithm tries to select the best algorithm automatically for each surface in the model. As of Gmsh 2.7, the "Automatic" algorithm selects "Delaunay" for plane surfaces and "MeshAdapt" for all other surfaces.

In 3D two unstructured algorithms are available:

1. The "Delaunay" algorithm is split into two separate steps. First, an initial mesh of the union of all the volumes in the model is performed using H. Si's Tetgen algorithm[6]. Then a three-dimensional version of the 2D Delaunay algorithm described above is applied.
2. The "Frontal" algorithm uses J. Schoeberl's Netgen algorithm[7].

The "Delaunay" algorithm is the most robust and the fastest, and is the only one that supports the `Field` mechanism to specify element sizes (see Section 6.3.1 [Specifying mesh element sizes], page 41). However, this algorithm will sometimes modify the surface mesh, and is thus not suitable for producing hybrid structured/unstructured grids. In that case the "Frontal" algorithm should be preferred. The quality of the elements produced by both algorithms is comparable. If element quality is important the mesh optimizer(s) should be applied.

6.2 Elementary vs. physical entities

If only elementary geometrical entities are defined (or if the `Mesh.SaveAll` option is set; see Section B.3 [Mesh options list], page 163), the grid produced by the mesh module will be saved "as is". That is, all the elements in the grid will be saved using the identification number of the elementary entities they discretize as their elementary region number (and 0 as their physical region number[8]; Chapter 9 [File formats], page 87). This can sometimes be inconvenient:

- mesh elements cannot be duplicated;
- the orientation of the mesh elements (the ordering of their nodes) is determined entirely by the orientation of their "parent" elementary entities, and cannot be modified;
- elements belonging to different elementary entities cannot be linked as being part of a larger group having a physical or mathematical meaning (like 'Left wing', 'Metallic part', 'Dirichlet boundary condition', ...).

To remedy these problems, the geometry module (see Chapter 5 [Geometry module], page 31) introduces the notion of "physical" entities (also called "physical groups").

[6] H. Si, *Tetgen: a quality tetrahedral mesh generator and three-dimensional Delaunay triangulator*, 2004.

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

[8] This behaviour was introduced in Gmsh 2.0. In older versions, both the elementary and the physical region numbers would be set to the identification number of the elementary region.

The purpose of physical entities is to assemble elementary entities into larger, possibly overlapping groups, and to control the orientation of the elements in these groups. The introduction of physical entities in large models usually greatly facilitates the manipulation of the model (e.g., using 'Tools->Visibility' in the GUI) and the interfacing with external solvers.

In the MSH file format (see Chapter 9 [File formats], page 87), if physical entities are defined, the output mesh only contains those elements that belong to physical entities. Other file formats each treat physical entities in slightly different ways, depending on their capability to define groups.

In all cases, Gmsh reindexes the mesh vertices and elements so that they are numbered in a continuous sequence in the output files. Note that the numbers displayed on screen after mesh generation thus usually differ from the ones saved in the mesh files. To check the actual numbers saved in the output file just load the file back using 'File->Open'.

6.3 Mesh commands

The mesh module commands mostly permit to modify the mesh element sizes and specify structured grid parameters. The actual mesh "actions" (i.e., "mesh the lines", "mesh the surfaces" and "mesh the volumes") cannot be specified in the script files. They have to be given either in the GUI or on the command line (see Chapter 3 [Running Gmsh on your system], page 11, and Section 3.3 [Command-line options], page 12).

6.3.1 Specifying mesh element sizes

There are three ways to specify the size of the mesh elements for a given geometry:

1. First, if `Mesh.CharacteristicLengthFromPoints` is set (it is by default), you can simply specify desired mesh element sizes at the geometrical points of the model (with the `Point` command: see Section 5.1.1 [Points], page 31). The size of the mesh elements will then be computed by linearly interpolating these values on the initial mesh (see Section 1.2 [Mesh], page 5). This might sometimes lead to over-refinement in some areas, so that you may have to add "dummy" geometrical entities in the model in order to get the desired element sizes.

   This method works with all the algorithms implemented in the mesh module. The final element sizes are of course constrained by the structured algorithms for which the element sizes are explicitly specified (e.g., transfinite and extruded grids: see Section 6.3.2 [Structured grids], page 55).

2. Second, if `Mesh.CharacteristicLengthFromCurvature` is set (it is not by default), the mesh will be adapted with respect to the curvature of the geometrical entities.

3. Finally, you can specify general mesh size "fields". Various fields exist:

   - A `PostView` field specifies an explicit background mesh in the form of a scalar post-processing view (see Section 8.1 [Post-processing commands], page 62, and Chapter 9 [File formats], page 87) in which the nodal values are the target element sizes. This method is very general but it requires a first (usually rough) mesh and a way to compute the target sizes on this mesh (usually through an error estimation procedure, in an iterative process of mesh adaptation). Warning: only parsed ('`.pos`') files can currently be used as background meshes ('`.msh`' files cannot be used, since the mesh used to define the field will be destroyed during the meshing process).

(Note that you can also load a background mesh directly from the command line using the `-bgm` option (see Section 3.3 [Command-line options], page 12), or in the GUI by selecting 'Apply as background mesh' in the post-processing view option menu.)

- A `Box` field specifies the size of the elements inside and outside of a parallelipipedic region.
- A `Threshold` field specifies the size of the mesh according to the distance to some geometrical entities. These entities can for example be geometry points and lines specified by an `Attractor` field.
- A `MathEval` field specifies the size of the mesh using an explicit mathematical function.
- A `Min` field specifies the size as the minimum of the sizes computed using other fields
- ...

Fields are supported by all the algorithms except those based on Netgen. The list of available fields with their options is given below.

The three aforementioned methods can be used simultaneously, in which case the smallest element size is selected at any given point.

All element sizes are further constrained by the `Mesh.CharacteristicLengthMin`, `Mesh.CharacteristicLengthMax` and `Mesh.CharacteristicLengthFactor` options (see Section B.3 [Mesh options list], page 163)

Here are the mesh commands that are related to the specification of mesh element sizes:

`Characteristic Length {` *`expression-list`* `} =` *`expression`*`;`

Modify the prescribed mesh element size of the points whose identification numbers are listed in *expression-list*. The new value is given by *expression*.

`Field[`*`expression`*`] =` *`string`*`;`

Create a new field (with id number *expression*), of type *string*.

`Field[`*`expression`*`].`*`string`* `=` *`char-expression`* `|` *`expression`* `|` *`expression-list`*`;`

Set the option *string* of the *expression*-th field.

`Background Field =` *`expression`*`;`

Select the *expression*-th field as the one used to compute element sizes. Only one background field can be given; if you want to combine several field, use the `Min` or `Max` field (see below).

Here is the list of all available fields with their associated options:

`Attractor`

Compute the distance from the nearest node in a list. It can also be used to compute the distance from curves, in which case each curve is replaced by NNodesByEdge equidistant nodes and the distance from those nodes is computed. Options:

`EdgesList`
: Indices of curves in the geometric model
type: list
default value: `{}`

`FacesList`
: Indices of surfaces in the geometric model (Warning, this feature is still experimental. It might (read: will probably) give wrong results for complex surfaces)
type: list
default value: `{}`

`FieldX`
: Id of the field to use as x coordinate.
type: integer
default value: `-1`

`FieldY`
: Id of the field to use as y coordinate.
type: integer
default value: `-1`

`FieldZ`
: Id of the field to use as z coordinate.
type: integer
default value: `-1`

`NNodesByEdge`
: Number of nodes used to discretized each curve
type: integer
default value: `20`

`NodesList`
: Indices of nodes in the geometric model
type: list
default value: `{}`

`AttractorAnisoCurve`
: Compute the distance from the nearest curve in a list. Then the mesh size can be specified independently in the direction normal to the curve and in the direction parallel to the curve (Each curve is replaced by NNodesByEdge equidistant nodes and the distance from those nodes is computed.)
Options:

`EdgesList`
: Indices of curves in the geometric model
type: list
default value: `{}`

`NNodesByEdge`
: Number of nodes used to discretized each curve
type: integer
default value: `20`

`dMax` Maxmium distance, above this distance from the curves, prescribe the maximum mesh sizes.
type: float
default value: `0.5`

`dMin` Minimum distance, bellow this distance from the curves, prescribe the minimum mesh sizes.
type: float
default value: `0.1`

`lMaxNormal`
Maximum mesh size in the direction normal to the closest curve.
type: float
default value: `0.5`

`lMaxTangent`
Maximum mesh size in the direction tangeant to the closest curve.
type: float
default value: `0.5`

`lMinNormal`
Minimum mesh size in the direction normal to the closest curve.
type: float
default value: `0.05`

`lMinTangent`
Minimum mesh size in the direction tangeant to the closest curve.
type: float
default value: `0.5`

`BoundaryLayer`
hwall * ratio^(dist/hwall)
Options:

`AnisoMax` Threshold angle for creating a mesh fan in the boundary layer
type: float
default value: `10000000000`

`EdgesList`
Indices of curves in the geometric model for which a boundary layer is needed
type: list
default value: `{}`

`FacesList`
Indices of faces in the geometric model for which a boundary layer is needed
type: list
default value: `{}`

`IntersectMetrics`
: Intersect metrics of all faces
type: integer
default value: `0`

`NodesList`
: Indices of nodes in the geometric model
type: list
default value: `{}`

`Quads`
: Generate recombined elements in the boundary layer
type: integer
default value: `0`

`fan_angle`
: Threshold angle for creating a mesh fan in the boundary layer
type: float
default value: `30`

`hfar`
: Element size far from the wall
type: float
default value: `1`

`hwall_n`
: Mesh Size Normal to the The Wall
type: float
default value: `0.1`

`hwall_t`
: Mesh Size Tangent to the Wall
type: float
default value: `0.5`

`ratio`
: Size Ratio Between Two Successive Layers
type: float
default value: `1.1`

`thickness`
: Maximal thickness of the boundary layer
type: float
default value: `0.01`

`Box`
: The value of this field is VIn inside the box, VOut outside the box. The box is given by

Xmin <= x <= XMax &&
YMin <= y <= YMax &&
ZMin <= z <= ZMax
Options:

`VIn`
: Value inside the box
type: float
default value: `0`

`VOut`
: Value outside the box
  type: float
  default value: `0`

`XMax`
: Maximum X coordinate of the box
  type: float
  default value: `0`

`XMin`
: Minimum X coordinate of the box
  type: float
  default value: `0`

`YMax`
: Maximum Y coordinate of the box
  type: float
  default value: `0`

`YMin`
: Minimum Y coordinate of the box
  type: float
  default value: `0`

`ZMax`
: Maximum Z coordinate of the box
  type: float
  default value: `0`

`ZMin`
: Minimum Z coordinate of the box
  type: float
  default value: `0`

`Centerline`

The value of this field is the distance to the centerline.

You should specify a fileName that contains the centerline. The centerline of a surface can be obtained with the open source software vmtk (http://www.vmtk.org/) using the following script:

vmtk vmtkcenterlines -seedselector openprofiles -ifile mysurface.stl -ofile centerlines.vtp –pipe vmtksurfacewriter -ifile centerlines.vtp -ofile centerlines.vtk

Options:

`FileName`
: File name for the centerlines
  type: string
  default value: `"centerlines.vtk"`

`closeVolume`
: Action: Create In/Outlet planar faces
  type: integer
  default value: `0`

`extrudeWall`
Action: Extrude wall
type: integer
default value: `0`

`hLayer` Thickness (% of radius) of the extruded layer
type: float
default value: `0.3`

`hSecondLayer`
Thickness (% of radius) of the second extruded layer
type: float
default value: `0.3`

`nbElemLayer`
Number of mesh elements the extruded layer
type: integer
default value: `3`

`nbElemSecondLayer`
Number of mesh elements the second extruded layer
type: integer
default value: `0`

`nbPoints` Number of mesh elements in a circle
type: integer
default value: `25`

`reMesh` Action: Cut the initial mesh in different mesh partitions using the centerlines
type: integer
default value: `0`

Actions:

`run` Run actions (closeVolume, extrudeWall, cutMesh)

`Curvature`

Compute the curvature of Field[IField]:

F = div(norm(grad(Field[IField])))
Options:

`Delta` Step of the finite differences
type: float
default value: `0`

`IField` Field index
type: integer
default value: `1`

`Cylinder` The value of this field is VIn inside a frustrated cylinder, VOut outside. The cylinder is given by

||dX||^2 < R^2 &&
(X-X0).A < ||A||^2
dX = (X - X0) - ((X - X0).A)/(||A||^2) . A
Options:

`Radius`	Radius type: float default value: `0`
`VIn`	Value inside the cylinder type: float default value: `0`
`VOut`	Value outside the cylinder type: float default value: `0`
`XAxis`	X component of the cylinder axis type: float default value: `0`
`XCenter`	X coordinate of the cylinder center type: float default value: `0`
`YAxis`	Y component of the cylinder axis type: float default value: `0`
`YCenter`	Y coordinate of the cylinder center type: float default value: `0`
`ZAxis`	Z component of the cylinder axis type: float default value: `1`
`ZCenter`	Z coordinate of the cylinder center type: float default value: `0`

`Frustum` This field is an extended cylinder with inner (i) and outer (o) radiuseson both endpoints (1 and 2). Length scale is bilinearly interpolated betweenthese locations (inner and outer radiuses, endpoints 1 and 2)The field values for a point P are given by : u = P1P.P1P2/||P1P2|| r = || P1P - u*P1P2 || Ri = (1-u)*R1i + u*R2i Ro = (1-u)*R1o + u*R2o v = (r-Ri)/(Ro-Ri) lc = (1-v)*((1-u)*v1i + u*v2i) + v*((1-u)*v1o + u*v2o) where (u,v) in [0,1]x[0,1]
Options:

`R1_inner`	Inner radius of Frustum at endpoint 1 type: float default value: `0`
`R1_outer`	Outer radius of Frustum at endpoint 1 type: float default value: `1`
`R2_inner`	Inner radius of Frustum at endpoint 2 type: float default value: `0`
`R2_outer`	Outer radius of Frustum at endpoint 2 type: float default value: `1`
`V1_inner`	Element size at point 1, inner radius type: float default value: `0.1`
`V1_outer`	Element size at point 1, outer radius type: float default value: `1`
`V2_inner`	Element size at point 2, inner radius type: float default value: `0.1`
`V2_outer`	Element size at point 2, outer radius type: float default value: `1`
`X1`	X coordinate of endpoint 1 type: float default value: `0`
`X2`	X coordinate of endpoint 2 type: float default value: `0`
`Y1`	Y coordinate of endpoint 1 type: float default value: `0`
`Y2`	Y coordinate of endpoint 2 type: float default value: `0`
`Z1`	Z coordinate of endpoint 1 type: float default value: `1`
`Z2`	Z coordinate of endpoint 2 type: float default value: `5.98152627002258e-154`

`Gradient` Compute the finite difference gradient of Field[IField]:

F = (Field[IField](X + Delta/2) - Field[IField](X - Delta/2)) / Delta
Options:

`Delta` Finite difference step
type: float
default value: `0`

`IField` Field index
type: integer
default value: `1`

`Kind` Component of the gradient to evaluate: 0 for X, 1 for Y, 2 for Z, 3 for the norm
type: integer
default value: `0`

`Laplacian`
Compute finite difference the Laplacian of Field[IField]:

F = G(x+d,y,z) + G(x-d,y,z) +
G(x,y+d,z) + G(x,y-d,z) +
G(x,y,z+d) + G(x,y,z-d) - 6 * G(x,y,z),

where G=Field[IField] and d=Delta
Options:

`Delta` Finite difference step
type: float
default value: `0.1`

`IField` Field index
type: integer
default value: `1`

`LonLat` Evaluate Field[IField] in geographic coordinates (longitude, latitude):

F = Field[IField](atan(y/x), asin(z/sqrt(x^2+y^2+z^2))
Options:

`FromStereo`
if = 1, the mesh is in stereographic coordinates. xi = 2Rx/(R+z), eta = 2Ry/(R+z)
type: integer
default value: `0`

`IField` Index of the field to evaluate.
type: integer
default value: `1`

`RadiusStereo`
: radius of the sphere of the stereograpic coordinates
  type: float
  default value: `6371000`

`MathEval`
: Evaluate a mathematical expression. The expression can contain x, y, z for spatial coordinates, F0, F1, ... for field values, and and mathematical functions.
  Options:

  `F`
  : Mathematical function to evaluate.
    type: string
    default value: `"F2 + Sin(z)"`

  Actions:

  `test`
  : description blabla

`MathEvalAniso`
: Evaluate a metric expression. The expressions can contain x, y, z for spatial coordinates, F0, F1, ... for field values, and and mathematical functions.
  Options:

  `m11`
  : element 11 of the metric tensor.
    type: string
    default value: `"F2 + Sin(z)"`

  `m12`
  : element 12 of the metric tensor.
    type: string
    default value: `"F2 + Sin(z)"`

  `m13`
  : element 13 of the metric tensor.
    type: string
    default value: `"F2 + Sin(z)"`

  `m22`
  : element 22 of the metric tensor.
    type: string
    default value: `"F2 + Sin(z)"`

  `m23`
  : element 23 of the metric tensor.
    type: string
    default value: `"F2 + Sin(z)"`

  `m33`
  : element 33 of the metric tensor.
    type: string
    default value: `"F2 + Sin(z)"`

`Max`
: Take the maximum value of a list of fields.
  Options:

`FieldsList`
Field indices
type: list
default value: `{}`

`MaxEigenHessian`
Compute the maximum eigenvalue of the Hessian matrix of Field[IField], with the gradients evaluated by finite differences:

F = max(eig(grad(grad(Field[IField]))))
Options:

`Delta` Step used for the finite differences
type: float
default value: `0`

`IField` Field index
type: integer
default value: `1`

`Mean` Simple smoother:

F = (G(x+delta,y,z) + G(x-delta,y,z) +
G(x,y+delta,z) + G(x,y-delta,z) +
G(x,y,z+delta) + G(x,y,z-delta) +
G(x,y,z)) / 7,

where G=Field[IField]
Options:

`Delta` Distance used to compute the mean value
type: float
default value: `0.0001`

`IField` Field index
type: integer
default value: `0`

`Min` Take the minimum value of a list of fields.
Options:

`FieldsList`
Field indices
type: list
default value: `{}`

`MinAniso` Take the intersection of a list of possibly anisotropic fields.
Options:

`FieldsList`
: Field indices
  type: list
  default value: `{}`

`Param` Evaluate Field IField in parametric coordinates:

F = Field[IField](FX,FY,FZ)

See the MathEval Field help to get a description of valid FX, FY and FZ expressions.
Options:

`FX` X component of parametric function
type: string
default value: `""`

`FY` Y component of parametric function
type: string
default value: `""`

`FZ` Z component of parametric function
type: string
default value: `""`

`IField` Field index
type: integer
default value: `1`

`PostView` Evaluate the post processing view IView.
Options:

`CropNegativeValues`
return LC_MAX instead of a negative value (this option is needed for backward compatibility with the BackgroundMesh option
type: boolean
default value: `1`

`IView` Post-processing view index
type: integer
default value: `0`

`Restrict` Restrict the application of a field to a given list of geometrical curves, surfaces or volumes.
Options:

`EdgesList`
Curve indices
type: list
default value: `{}`

`FacesList`
: Surface indices
  type: list
  default value: `{}`

`IField`
: Field index
  type: integer
  default value: `1`

`RegionsList`
: Volume indices
  type: list
  default value: `{}`

`Structured`

Linearly interpolate between data provided on a 3D rectangular structured grid.

The format of the input file is:

Ox Oy Oz
Dx Dy Dz
nx ny nz
v(0,0,0) v(0,0,1) v(0,0,2) ...
v(0,1,0) v(0,1,1) v(0,1,2) ...
v(0,2,0) v(0,2,1) v(0,2,2) ...
... ...
v(1,0,0) ... ...

where O are the coordinates of the first node, D are the distances between nodes in each direction, n are the numbers of nodes in each direction, and v are the values on each node.
Options:

`FileName`
: Name of the input file
  type: path
  default value: `""`

`TextFormat`
: True for ASCII input files, false for binary files (4 bite signed integers for n, double precision floating points for v, D and O)
  type: boolean
  default value: `0`

`Threshold`

F = LCMin if Field[IField] <= DistMin,
F = LCMax if Field[IField] >= DistMax,
F = interpolation between LcMin and LcMax if DistMin < Field[IField] < DistMax
Options:

`DistMax` Distance from entity after which element size will be LcMax
type: float
default value: `10`

`DistMin` Distance from entity up to which element size will be LcMin
type: float
default value: `1`

`IField` Index of the field to evaluate
type: integer
default value: `0`

`LcMax` Element size outside DistMax
type: float
default value: `1`

`LcMin` Element size inside DistMin
type: float
default value: `0.1`

`Sigmoid` True to interpolate between LcMin and LcMax using a sigmoid, false to interpolate linearly
type: boolean
default value: `0`

`StopAtDistMax`
True to not impose element size outside DistMax (i.e., F = a very big value if Field[IField] > DistMax)
type: boolean
default value: `0`

`UTM` Evaluate Field[IField] in Universal Transverse Mercator coordinates.

The formulas for the coordinates transformation are taken from:

http://www.uwgb.edu/dutchs/UsefulData/UTMFormulas.HTM
Options:

`IField` Index of the field to evaluate
type: integer
default value: `1`

`Zone` Zone of the UTM projection
type: integer
default value: `0`

6.3.2 Structured grids

`Extrude {` *`expression-list`* `} {` *`extrude-list layers`* `}`
Extrudes both the geometry and the mesh using a translation (see Section 5.1.5 [Extrusions], page 35). The *layers* option determines how the mesh is extruded and has the following syntax:

```
layers:
  Layers { expression } |
  Layers { { expression-list }, { expression-list } } |
  Recombine; ...
```

In the first `Layers` form, *expression* gives the number of elements to be created in the (single) layer. In the second form, the first *expression-list* defines how many elements should be created in each extruded layer, and the second *expression-list* gives the normalized height of each layer (the list should contain a sequence of *n* numbers $0 < h1 < h2 < \ldots < hn <= 1$). See Section A.3 [t3.geo], page 109, for an example.

For line extrusions, the `Recombine` option will recombine triangles into quadrangles when possible. For surface extrusions, the `Recombine` option will recombine tetrahedra into prisms, hexahedra or pyramids.

Please note that, starting with Gmsh 2.0, region numbers cannot be specified explicitly anymore in `Layers` commands. Instead, as with all other geometry commands, you must use the automatically created entity identifier created by the extrusion command. For example, the following extrusion command will return the id of the new "top" surface in `num[0]` and the id of the new volume in `num[1]`:

```
num[] = Extrude {0,0,1} { Surface{1}; Layers{10}; };
```

`Extrude { {` *`expression-list`* `}, {` *`expression-list`* `},` *`expression`* `} {` *`extrude-list layers`* `}`

Extrudes both the geometry and the mesh using a rotation (see Section 5.1.5 [Extrusions], page 35). The *layers* option is defined as above.

`Extrude { {` *`expression-list`* `}, {` *`expression-list`* `}, {` *`expression-list`* `},` *`expression`* `} {` *`extrude-list layers`* `}`

Extrudes both the geometry and the mesh using a combined translation and rotation (see Section 5.1.5 [Extrusions], page 35). The *layers* option is defined as above.

`Extrude { Surface {` *`expression-list`* `};` *`layers`* `< Using Index[`*`expr`*`]; > < Using View[`*`expr`*`]; > }`

Extrudes a boundary layer from the specified surfaces. If no view is specified, the boundary layer is created using gouraud-shaped (smoothed) normal field. Specifying a boundary layer index allows to extrude several independent boundary layers (with independent normal smoothing).

`Transfinite Line {` *`expression-list`* `} | "*" =` *`expression`* `< Using Progression | Bump` *`expression`* `>;`

Selects the lines in *expression-list* to be meshed with the 1D transfinite algorithm. The *expression* on the right hand side gives the number of nodes that will be created on the line (this overrides any other mesh element size prescription—see Section 6.3.1 [Specifying mesh element sizes], page 41). The optional argument '`Using Progression` *`expression`*' instructs the transfinite algorithm to distribute the nodes following a geometric progression (`Progression 2` meaning for example that each line element in the series will be twice as long as the preceding one). The optional argument '`Using Bump` *`expression`*' instructs the transfinite algorithm to distribute the nodes with a refinement at both ends of the line.

`Transfinite Surface {` *`expression-list`* `} | "*" < = {` *`expression-list`* `} > < Left | Right | Alternate > ;`

Selects surfaces to be meshed with the 2D transfinite algorithm. The *expression-list* on the right-hand-side should contain the identification numbers of three or four points on the boundary of the surface that define the corners of the transfinite interpolation. If no identification numbers are given, the transfinite algorithm will try to find the corners automatically. The optional argument specifies the way the triangles are oriented when the mesh is not recombined.

`Transfinite Volume {` *`expression-list`* `} | "*" < = {` *`expression-list`* `} > ;`

Selects five- or six-face volumes to be meshed with the 3D transfinite algorithm. The *expression-list* on the right-hand-side should contain the identification numbers of the six or eight points on the boundary of the volume that define the corners of the transfinite interpolation. If no identification numbers are given, the transfinite algorithm will try to find the corners automatically.

6.3.3 Miscellaneous

Here is a list of all other mesh commands currently available:

`Point | Line {` *`expression-list`* `} In Surface {` *`expression`* `};`

Embed the point(s) or line(s) in the given surface. The surface mesh will conform to the mesh of the point(s) or lines(s). Note that embedded lines only work with the MeshAdapt 2-D algorithm.

`Periodic Line {` *`expression-list`* `} = {` *`expression-list`* `};`

Force mesh of lines on the left-hand side (slaves) to match the mesh of the lines on the right-hand side (masters).

`Periodic Surface` *`expression`* `{` *`expression-list`* `} =` *`expression`* `{` *`expression-list`* `};`

Force mesh of the surface on the left-hand side (slave, with boundary edges specified between braces) to match the mesh of the surface on the right-hand side (master, with boundary edges specified between braces).

`Coherence Mesh;`

Removes all duplicate mesh vertices.

`Color` *`color-expression`* `{ Point | Line | Surface | Volume {` *`expression-list`* `}; ... }`

Sets the mesh color of the entities in *expression-list* to *color-expression*.

`Hide { Point | Line | Surface | Volume {` *`expression-list`* `}; ... }`

Hides the mesh of the entities in *expression-list*, if `General.VisibilityMode` is set to `0` or `2`.

`Hide` *`char-expression`* `;`

Hides the mesh of the entity *char-expression*, if `General.VisibilityMode` is set to `0` or `2` (*char-expression* can for example be `"*"`).

`Recombine Surface {` *expression-list* `} | "*" < =` *expression* `>;`
: Recombines the triangular meshes of the surfaces listed in *expression-list* into mixed triangular/quadrangular meshes. The optional *expression* on the right hand side specifies the maximum difference (in degrees) allowed between the largest angle of a quadrangle and a right angle (a value of 0 would only accept quadrangles with right angles; a value of 90 would allow degenerate quadrangles; default value is 45).

`Save` *char-expression* `;`
: Saves the mesh in a file named *char-expression*, using the current `Mesh.Format` (see Section B.3 [Mesh options list], page 163). If the path in *char-expression* is not absolute, *char-expression* is appended to the path of the current file.

`Show { Point | Line | Surface | Volume {` *expression-list* `}; ... }`
: Shows the mesh of the entities in *expression-list*, if `General.VisibilityMode` is set to `0` or `2`.

`Show` *char-expression* `;`
: Shows the mesh of the entity *char-expression*, if `General.VisibilityMode` is set to `0` or `2` (*char-expression* can for example be `"*"`).

`Smoother Surface {` *expression-list* `} =` *expression* `;`
: Sets number of elliptic smoothing steps for the surfaces listed in *expression-list* (smoothing only applies to transfinite meshes at the moment).

`Homology ( {` *expression-list* `} ) { {` *expression-list* `} , {` *expression-list* `} };`
: Compute a basis representation for homology spaces after a mesh has been generated. The first *expression-list* is a list of dimensions whose homology bases are computed; if empty, all bases are computed. The second *expression-list* is a list physical groups that constitute the computation domain; if empty, the whole mesh is the domain. The third *expression-list* is a list of physical groups that constitute the relative subdomain of relative homology computation; if empty, absolute homology is computed. Resulting basis representation chains are stored as physical groups in the mesh.

`Cohomology ( {` *expression-list* `} ) { {` *expression-list* `} , {` *expression-list* `}`
`};`
: Similar to command `Homology`, but computes a basis representation for cohomology spaces instead.

6.4 Mesh options

The list of all the options that control the behavior of mesh commands, as well as the way meshes are displayed in the GUI, is given in Section B.3 [Mesh options list], page 163.

7 Solver module

External solvers can be driven by Gmsh through the ONELAB `http://www.onelab.info` interface. To add a new solver in the solver module, you need to specify its name (`Solver.Name0`, `Solver.Name1`, etc.) and the path to the executable (`Solver.Executable0`, `Solver.Executable1`, etc.); see Section B.4 [Solver options list], page 179).

The client-server API for the solver interface is defined in the '`onelab.h`' header. See '`utils/solvers/c++/solver.cpp`' for a simple example on how to use the ONELAB programming interface. See the sources of GetDP (`http://geuz.org/getdp` for a more comprehensive example.

7.1 Solver options

The list of all the solver options is given in Section B.4 [Solver options list], page 179.

8 Post-processing module

Gmsh's post-processing module can handle multiple scalar, vector or tensor datasets along with the geometry and the mesh. The datasets can be given in several formats: in human-readable "parsed" format (these are just part of a standard input script, but are usually put in separate files with a '`.pos`' extension), in native MSH files (ASCII or binary files with '`.msh`' extensions: see Chapter 9 [File formats], page 87), or in standard third-party formats (like MED: `http://www.code-aster.org/outils/med/`).

Once loaded into Gmsh, scalar fields can be displayed as iso-value lines and surfaces or color maps, whereas vector fields can be represented either by three-dimensional arrows or by displacement maps. (Tensor fields are currently displayed as Von-Mises effective stresses, min/max eigenvalues, eigenvectors, ellipsis or ellipsoid. To display other (combinations of) components, you can use the `Force scalar` or `Force vector` options, or use `Plugin(MathEval)`: see Section 8.2 [Post-processing plugins], page 65.)

In Gmsh's jargon, each dataset is called a "view". Each view is given a name, and can be manipulated either individually (each view has its own button in the GUI and can be referred to by its index in a script) or globally (see the `PostProcessing.Link` option in Section B.5 [Post-processing options list], page 182).

By default, Gmsh treats all post-processing views as three-dimensional plots, i.e., draws the scalar, vector and tensor primitives (points, lines, triangles, tetrahedra, etc.) in 3D space. But Gmsh can also represent each post-processing view containing *scalar points* as two-dimensional ("X-Y") plots, either space- or time-oriented:

- in a '2D space' plot, the scalar points are taken in the same order as they are defined in the post-processing view: the abscissa of the 2D graph is the curvilinear abscissa of the curve defined by the point series, and only one curve is drawn using the values associated with the points. If several time steps are available, each time step generates a new curve;
- in a '2D time' plot, one curve is drawn for each scalar point in the view and the abscissa is the time step.

Although visualization is usually mostly an interactive task, Gmsh exposes all the post-processing commands and options to the user in its scripting language to permit a complete automation of the post-processing process (see e.g., Section A.8 [t8.geo], page 118, and Section A.9 [t9.geo], page 121).

The two following sections summarize all available post-processing commands and options. Most options apply to both 2D and 3D plots (colormaps, point/line sizes, interval types, time step selection, etc.), but some are peculiar to 3D (lightning, element selection, etc.) or 2D plots (abscissa labels, etc.). Note that 2D plots can be positioned explicitly inside the graphical window, or be automatically positioned in order to avoid overlaps.

Sample post-processing files in human-readable "parsed" format and in the native MSH file format are available in the '`tutorial`' directory of Gmsh's distribution ('`.pos`' and '`.msh`' files). The "parsed" format is defined in the next section (cf. the `View` command); the MSH format is defined in Chapter 9 [File formats], page 87.

8.1 Post-processing commands

`Alias View[`*`expression`*`];`

Creates an alias of the *expression*-th post-processing view.

Note that `Alias` creates a logical duplicate of the view without actually duplicating the data in memory. This is very useful when you want multiple simultaneous renderings of the same large dataset (usually with different display options), but you cannot afford to store all copies in memory. If what you really want is multiple physical copies of the data, just merge the file containing the post-processing view multiple times.

`AliasWithOptions View[`*`expression`*`];`

Creates an alias of the *expression*-th post-processing view and copies all the options of the *expression*-th view to the new aliased view.

`Combine ElementsByViewName;`

Combines all the post-processing views having the same name into new views. The combination is done “spatially”, i.e., simply by appending the elements at the end of the new views.

`Combine ElementsFromAllViews | Combine Views;`

Combines all the post-processing views into a single new view. The combination is done “spatially”, i.e., simply by appending the elements at the end of the new view.

`Combine ElementsFromVisibleViews;`

Combines all the visible post-processing views into a single new view. The combination is done “spatially”, i.e., simply by appending the elements at the end of the new view.

`Combine TimeStepsByViewName | Combine TimeSteps;`

Combines the data from all the post-processing views having the same name into new multi-time-step views. The combination is done “temporally”, i.e., as if the data in each view corresponds to a different time instant. The combination will fail if the meshes in all the views are not identical.

`Combine TimeStepsFromAllViews;`

Combines the data from all the post-processing views into a new multi-time-step view. The combination is done “temporally”, i.e., as if the data in each view corresponds to a different time instant. The combination will fail if the meshes in all the views are not identical.

`Combine TimeStepsFromVisibleViews;`

Combines the data from all the visible post-processing views into a new multi-time-step view. The combination is done “temporally”, i.e., as if the data in each view corresponds to a different time instant. The combination will fail if the meshes in all the views are not identical.

`Delete View[`*`expression`*`];`

Deletes (removes) the *expression*-th post-processing view. Note that post-processing view numbers start at 0.

`Delete Empty Views;`
: Deletes (removes) all the empty post-processing views.

`Background Mesh View[`*`expression`*`];`
: Applies the *expression*-th post-processing view as the current background mesh. Note that post-processing view numbers start at 0.

`Plugin (`*`string`*`) . Run;`
: Executes the plugin *string*. The list of default plugins is given in Section 8.2 [Post-processing plugins], page 65.

`Plugin (`*`string`*`) .` *`string`* `=` *`expression`* `|` *`char-expression`*`;`
: Sets an option for a given plugin. See Section 8.2 [Post-processing plugins], page 65, for a list of default plugins and Section A.9 [t9.geo], page 121, for some examples.

`Save View[`*`expression`*`]` *`char-expression`*`;`
: Saves the the *expression*-th post-processing view in a file named *char-expression*. If the path in *char-expression* is not absolute, *char-expression* is appended to the path of the current file.

`View "`*`string`*`" {` *`string`* `< (` *`expression-list`* `) > {` *`expression-list`* `}; ... };`
: Creates a new post-processing view, named `"`*`string`*`"`. This is an easy and quite powerful way to import post-processing data: all the values are *expressions*, you can embed datasets directly into your geometrical descriptions (see, e.g., Section A.4 [t4.geo], page 111), the data can be easily generated "on-the-fly" (there is no header containing *a priori* information on the size of the dataset). The syntax is also very permissive, which makes it ideal for testing purposes.

  However this "parsed format" is read by Gmsh's script parser, which makes it inefficient if there are many elements in the dataset. Also, there is no connectivity information in parsed views and all the elements are independent (all fields can be discontinuous), so a lot of information can be duplicated. For large datasets, you should thus use the mesh-based post-processing file format described in Chapter 9 [File formats], page 87, or use one of the standard formats like MED.

  More explicitly, the syntax for a parsed `View` is the following

```
View "string" {
  type ( list-of-coords ) { list-of-values }; ...
  < TIME { expression-list }; >
  < INTERPOLATION_SCHEME { val-coef-matrix } { val-exp-matrix }
                         < { geo-coef-matrix } { geo-exp-matrix } > ; >
};
```

  where the 47 object *types* that can be displayed are:

```
                           type  #list-of-coords  #list-of-values
--------------------------------------------------------------------
Scalar point                SP    3               1  * nb-time-steps
Vector point                VP    3               3  * nb-time-steps
Tensor point                TP    3               9  * nb-time-steps
Scalar line                 SL    6               2  * nb-time-steps
Vector line                 VL    6               6  * nb-time-steps
```

`Tensor line`	`TL`	`6`	`18 *` *nb-time-steps*
`Scalar triangle`	`ST`	`9`	`3 *` *nb-time-steps*
`Vector triangle`	`VT`	`9`	`9 *` *nb-time-steps*
`Tensor triangle`	`TT`	`9`	`27 *` *nb-time-steps*
`Scalar quadrangle`	`SQ`	`12`	`4 *` *nb-time-steps*
`Vector quadrangle`	`VQ`	`12`	`12 *` *nb-time-steps*
`Tensor quadrangle`	`TQ`	`12`	`36 *` *nb-time-steps*
`Scalar tetrahedron`	`SS`	`12`	`4 *` *nb-time-steps*
`Vector tetrahedron`	`VS`	`12`	`12 *` *nb-time-steps*
`Tensor tetrahedron`	`TS`	`12`	`36 *` *nb-time-steps*
`Scalar hexahedron`	`SH`	`24`	`8 *` *nb-time-steps*
`Vector hexahedron`	`VH`	`24`	`24 *` *nb-time-steps*
`Tensor hexahedron`	`TH`	`24`	`72 *` *nb-time-steps*
`Scalar prism`	`SI`	`18`	`6 *` *nb-time-steps*
`Vector prism`	`VI`	`18`	`18 *` *nb-time-steps*
`Tensor prism`	`TI`	`18`	`54 *` *nb-time-steps*
`Scalar pyramid`	`SY`	`15`	`5 *` *nb-time-steps*
`Vector pyramid`	`VY`	`15`	`15 *` *nb-time-steps*
`Tensor pyramid`	`TY`	`15`	`45 *` *nb-time-steps*
`2D text`	`T2`	`3`	`arbitrary`
`3D text`	`T3`	`4`	`arbitrary`

The coordinates are given 'by node', i.e.,

- `(`*coord1*`,` *coord2*`,` *coord3*`)` for a point,
- `(`*coord1-node1*`,` *coord2-node1*`,` *coord3-node1*`,` *coord1-node2*`,` *coord2-node2*`,` *coord3-node2*`)` for a line,
- `(`*coord1-node1*`,` *coord2-node1*`,` *coord3-node1*`,` *coord1-node2*`,` *coord2-node2*`,` *coord3-node2*`,` *coord1-node3*`,` *coord2-node3*`,` *coord3-node3*`)` for a triangle,
- etc.

The ordering of the nodes is given in Section 9.3 [Node ordering], page 94.

The values are given by time step, by node and by component, i.e.:

```
comp1-node1-time1, comp2-node1-time1, comp3-node1-time1,
comp1-node2-time1, comp2-node2-time1, comp3-node2-time1,
comp1-node3-time1, comp2-node3-time1, comp3-node3-time1,
comp1-node1-time2, comp2-node1-time2, comp3-node1-time2,
comp1-node2-time2, comp2-node2-time2, comp3-node2-time2,
comp1-node3-time2, comp2-node3-time2, comp3-node3-time2,
...
```

For the 2D text objects, the two first *expression*s in *list-of-coords* give the X-Y position of the string in screen coordinates, measured from the top-left corner of the window. If the first (respectively second) *expression* is negative, the position is measured from the right (respectively bottom) edge of the window. If the value of the first (respectively second) *expression* is larger than 99999, the string is centered horizontally (respectively vertically). If the third *expression* is equal to zero, the text is aligned bottom-left and displayed using the default font and size. Otherwise, the third *expression* is converted into an integer whose eight lower bits give the font size, whose eight next bits select the font (the index corresponds to the position in the font menu in the GUI), and whose eight next bits define the text alignment (0=bottom-left, 1=bottom-center, 2=bottom-right, 3=top-left, 4=top-center, 5=top-right, 6=center-left, 7=center-center, 8=center-right).

For the 3D text objects, the three first *expressions* in *list-of-coords* give the XYZ position of the string in model (real world) coordinates. The fourth *expression* has the same meaning as the third *expression* in 2D text objects.

For both 2D and 3D text objects, the *list-of-values* can contain an arbitrary number of *char-expressions*.

The optional `TIME` list can contain a list of expressions giving the value of the time (or any other variable) for which an evolution was saved.

The optional `INTERPOLATION_SCHEME` lists can contain the interpolation matrices used for high-order adaptive visualization.

Let us assume that the approximation of the view's value over an element is written as a linear combination of *d* basis functions *f*[*i*], *i*=0, ..., *d*-1 (the coefficients being stored in *list-of-values*). Defining *f*[*i*] = Sum(*j*=0, ..., *d*-1) *F*[*i*][*j*] *p*[*j*], with *p*[*j*] = *u*^*P*[*j*][0] *v*^*P*[*j*][1] *w*^*P*[*j*][2] (*u*, *v* and *w* being the coordinates in the element's parameter space), then *val-coef-matrix* denotes the *d* x *d* matrix *F* and *val-exp-matrix* denotes the *d* x *3* matrix *P*.

In the same way, let us also assume that the coordinates *x*, *y* and *z* of the element are obtained through a geometrical mapping from parameter space as a linear combination of *m* basis functions *g*[*i*], *i*=0, ..., *m*-1 (the coefficients being stored in *list-of-coords*). Defining *g*[*i*] = Sum(*j*=0, ..., *m*-1) *G*[*i*][*j*] *q*[*j*], with *q*[*j*] = *u*^*Q*[*j*][0] *v*^*Q*[*j*][1] *w*^*Q*[*j*][2], then *val-coef-matrix* denotes the *m* x *m* matrix *G* and *val-exp-matrix* denotes the *m* x *3* matrix *Q*.

Here are for example the interpolation matrices for a first order quadrangle:

```
INTERPOLATION_SCHEME
{
  {1/4,-1/4, 1/4,-1/4},
  {1/4, 1/4,-1/4,-1/4},
  {1/4, 1/4, 1/4, 1/4},
  {1/4,-1/4,-1/4, 1/4}
}
{
  {0, 0, 0},
  {1, 0, 0},
  {0, 1, 0},
  {1, 1, 0}
};
```

8.2 Post-processing plugins

Post-processing plugins permit to extend the functionality of Gmsh's post-processing module. The difference between regular post-processing options (see Section B.5 [Post-processing options list], page 182) and post-processing plugins is that regular post-processing options only change the way the data is displayed, while post-processing plugins either create new post-processing views, or modify the data stored in a view (in a destructive, non-reversible way).

Plugins are available in the GUI by right-clicking on a view button (or by clicking on the black arrow next to the view button) and then selecting the 'Plugin' submenu.

Here is the list of the plugins that are shipped by default with Gmsh:

`Plugin(AnalyseCurvedMesh)`

Plugin(AnalyseCurvedMesh) check the jacobian of all elements of dimension 'Dim' or the greater model dimension if 'Dim' is either <0 or >3.

Analysis : 0 do nothing +1 find invalid elements (*) +2 compute J_min and J_max of all elements and print some statistics

Effect (for *) : 0 do nothing +1 print a list of invalid elements +2 print some statistics +4 hide valid elements (for GUI)

MaxDepth = 0,1,... 0 : only sample the jacobian 1 : compute Bezier coefficients 2+ : execute a maximum of 1+ subdivision(s)

JacBreak = [0,1[: if a value of the jacobian <= 'JacBreak' is found, the element is said to be invalid

BezBreak = [0,JacBreak[: if all Bezier coefficients are > 'BezBreak', the element is said to be valid

Tolerance = R+ , << 1 : tolerance (relatively to J_min and J_max) used during the computation of J_min and J_max Numeric options:

`Dim`	Default value: `-1`
`Analysis`	Default value: `2`
`Effect (1)`	Default value: `6`
`JacBreak (1)`	Default value: `0`
`BezBreak (1)`	Default value: `0`
`MaxDepth (1,2)`	Default value: `20`
`Tolerance (2)`	Default value: `0.001`

`Plugin(Annotate)`

Plugin(Annotate) adds the text string 'Text', in font 'Font' and size 'FontSize', in the view 'View'. The string is aligned according to 'Align'.

If 'ThreeD' is equal to 1, the plugin inserts the string in model coordinates at the position ('X','Y','Z'). If 'ThreeD' is equal to 0, the plugin inserts the string in screen coordinates at the position ('X','Y').

If 'View' < 0, the plugin is run on the current view.

Plugin(Annotate) is executed in-place for list-based datasets or creates a new view for other datasets. String options:

`Text`	Default value: `"My Text"`
`Font`	Default value: `"Helvetica"`
`Align`	Default value: `"Left"`

Numeric options:

`X`	Default value: `50`
`Y`	Default value: `30`
`Z`	Default value: `0`
`ThreeD`	Default value: `0`
`FontSize`	Default value: `14`
`View`	Default value: `-1`

`Plugin(Bubbles)`

Plugin(Bubbles) constructs a geometry consisting of 'bubbles' inscribed in the Voronoi of an input triangulation. 'ShrinkFactor' allows to change the size of the bubbles. The plugin expects a triangulation in the 'z = 0' plane to exist in the current model.

Plugin(Bubbles) creates one '.geo' file. String options:

`OutputFile`
Default value: `"bubbles.geo"`

Numeric options:

`ShrinkFactor`
Default value: `0`

`Plugin(Crack)`

Plugin(Crack) creates a crack around the physical group Numeric options:

`Dimension`
Default value: `1`

`PhysicalGroup`
Default value: `1`

`Plugin(Curl)`

Plugin(Curl) computes the curl of the field in the view 'View'.

If 'View' < 0, the plugin is run on the current view.

Plugin(Curl) creates one new view. Numeric options:

`View`	Default value: `-1`

`Plugin(CutBox)`

Plugin(CutBox) cuts the view 'View' with a rectangular box defined by the 4 points ('X0','Y0','Z0') (origin), ('X1','Y1','Z1') (axis of U), ('X2','Y2','Z2') (axis of V) and ('X3','Y3','Z3') (axis of W).

The number of points along U, V, W is set with the options 'NumPointsU', 'NumPointsV' and 'NumPointsW'.

If 'ConnectPoints' is zero, the plugin creates points; otherwise, the plugin generates hexahedra, quadrangles, lines or points depending on the values of 'NumPointsU', 'NumPointsV' and 'NumPointsW'.

If 'Boundary' is zero, the plugin interpolates the view inside the box; otherwise the plugin interpolates the view at its boundary.

If 'View' < 0, the plugin is run on the current view.

Plugin(CutBox) creates one new view. Numeric options:

`X0`	Default value: `0`
`Y0`	Default value: `0`
`Z0`	Default value: `0`
`X1`	Default value: `1`
`Y1`	Default value: `0`
`Z1`	Default value: `0`
`X2`	Default value: `0`
`Y2`	Default value: `1`
`Z2`	Default value: `0`
`X3`	Default value: `0`
`Y3`	Default value: `0`
`Z3`	Default value: `1`
`NumPointsU`	Default value: `20`
`NumPointsV`	Default value: `20`
`NumPointsW`	Default value: `20`
`ConnectPoints`	Default value: `1`
`Boundary`	Default value: `1`
`View`	Default value: `-1`

`Plugin(CutGrid)`

Plugin(CutGrid) cuts the view 'View' with a rectangular grid defined by the 3 points ('X0','Y0','Z0') (origin), ('X1','Y1','Z1') (axis of U) and ('X2','Y2','Z2') (axis of V).

The number of points along U and V is set with the options 'NumPointsU' and 'NumPointsV'.

If 'ConnectPoints' is zero, the plugin creates points; otherwise, the plugin generates quadrangles, lines or points depending on the values of 'NumPointsU' and 'NumPointsV'.

If 'View' < 0, the plugin is run on the current view.

Plugin(CutGrid) creates one new view. Numeric options:

`X0`	Default value: `0`
`Y0`	Default value: `0`
`Z0`	Default value: `0`
`X1`	Default value: `1`
`Y1`	Default value: `0`
`Z1`	Default value: `0`
`X2`	Default value: `0`
`Y2`	Default value: `1`
`Z2`	Default value: `0`
`NumPointsU`	Default value: `20`
`NumPointsV`	Default value: `20`
`ConnectPoints`	Default value: `1`
`View`	Default value: `-1`

`Plugin(CutParametric)`

Plugin(CutParametric) cuts the view 'View' with the parametric function ('X'(u,v), 'Y'(u,v), 'Z'(u,v)), using 'NumPointsU' values of the parameter u in ['MinU', 'MaxU'] and 'NumPointsV' values of the parameter v in ['MinV', 'MaxV'].

If 'ConnectPoints' is set, the plugin creates surface or line elements; otherwise, the plugin generates points.

If 'View' < 0, the plugin is run on the current view.

Plugin(CutParametric) creates one new view. String options:

`X`	Default value: `"2 * Cos(u) * Sin(v)"`
`Y`	Default value: `"4 * Sin(u) * Sin(v)"`
`Z`	Default value: `"0.1 + 0.5 * Cos(v)"`

Numeric options:

`MinU`	Default value: `0`
`MaxU`	Default value: `6.2832`
`NumPointsU`	Default value: `180`
`MinV`	Default value: `0`
`MaxV`	Default value: `6.2832`
`NumPointsV`	Default value: `180`

`ConnectPoints`
: Default value: `0`

`View`
: Default value: `-1`

`Plugin(CutPlane)`

Plugin(CutPlane) cuts the view 'View' with the plane 'A'*X + 'B'*Y + 'C'*Z + 'D' = 0.

If 'ExtractVolume' is nonzero, the plugin extracts the elements on one side of the plane (depending on the sign of 'ExtractVolume').

If 'View' < 0, the plugin is run on the current view.

Plugin(CutPlane) creates one new view. Numeric options:

`A`	Default value: `1`
`B`	Default value: `0`
`C`	Default value: `0`
`D`	Default value: `-0.01`
`ExtractVolume`	Default value: `0`
`RecurLevel`	Default value: `4`
`TargetError`	Default value: `0`
`View`	Default value: `-1`

`Plugin(CutSphere)`

Plugin(CutSphere) cuts the view 'View' with the sphere (X-'Xc')^2 + (Y-'Yc')^2 + (Z-'Zc')^2 = 'R'^2.

If 'ExtractVolume' is nonzero, the plugin extracts the elements inside (if 'ExtractVolume' < 0) or outside (if 'ExtractVolume' > 0) the sphere.

If 'View' < 0, the plugin is run on the current view.

Plugin(CutSphere) creates one new view. Numeric options:

`Xc`	Default value: `0`
`Yc`	Default value: `0`
`Zc`	Default value: `0`
`R`	Default value: `0.25`
`ExtractVolume`	Default value: `0`
`RecurLevel`	Default value: `4`
`TargetError`	Default value: `0`

`View` Default value: `-1`

`Plugin(DiscretizationError)`

Plugin(DiscretizationError) computes the error between the mesh and the geometry. It does so by supersampling the elements and computing the distance between the supersampled points dans their projection on the geometry. Numeric options:

`SuperSamplingNodes`
Default value: `10`

`Plugin(Distance)`

Plugin(Distance) computes distances to physical entities in a mesh.

Define the physical entities to which the distance is computed. If Point=0, Line=0, and Surface=0, then the distance is computed to all the boundaries of the mesh (edges in 2D and faces in 3D).

Computation<0. computes the geometrical euclidian distance (warning: different than the geodesic distance), and Computation=a>0.0 solves a PDE on the mesh with the diffusion constant mu = a*bbox, with bbox being the max size of the bounding box of the mesh (see paper Legrand 2006).

Min Scale and max Scale, scale the distance function. If min Scale<0 and max Scale<0, then no scaling is applied to the distance function.

Plugin(Distance) creates a new distance view and also saves the view in the fileName.pos file. String options:

`Filename` Default value: `"distance.pos"`

Numeric options:

`PhysPoint`
Default value: `0`

`PhysLine` Default value: `0`

`PhysSurface`
Default value: `0`

`Computation`
Default value: `-1`

`MinScale` Default value: `-1`

`MaxScale` Default value: `-1`

`Orthogonal`
Default value: `-1`

`Plugin(Divergence)`

Plugin(Divergence) computes the divergence of the field in the view 'View'.

If 'View' < 0, the plugin is run on the current view.

Plugin(Divergence) creates one new view. Numeric options:

`View` Default value: `-1`

`Plugin(Eigenvalues)`

Plugin(Eigenvalues) computes the three real eigenvalues of each tensor in the view 'View'.

If 'View' < 0, the plugin is run on the current view.

Plugin(Eigenvalues) creates three new scalar views. Numeric options:

`View`	Default value: `-1`

`Plugin(Eigenvectors)`

Plugin(Eigenvectors) computes the three (right) eigenvectors of each tensor in the view 'View' and sorts them according to the value of the associated eigenvalues.

If 'ScaleByEigenvalues' is set, each eigenvector is scaled by its associated eigenvalue. The plugin gives an error if the eigenvectors are complex.

If 'View' < 0, the plugin is run on the current view.

Plugin(Eigenvectors) creates three new vector view. Numeric options:

`ScaleByEigenvalues`	Default value: `1`
`View`	Default value: `-1`

`Plugin(ExtractEdges)`

Plugin(ExtractEdges) extracts sharp edges from a triangular mesh.

Plugin(ExtractEdges) creates one new view. Numeric options:

`Angle`	Default value: `40`
`IncludeBoundary`	Default value: `1`

`Plugin(ExtractElements)`

Plugin(ExtractElements) extracts some elements from the view 'View'. If 'MinVal' != 'MaxVal', it extracts the elements whose 'TimeStep'-th values (averaged by element) are comprised between 'MinVal' and 'MaxVal'. If 'Visible' != 0, it extracts visible elements.

If 'View' < 0, the plugin is run on the current view.

Plugin(ExtractElements) creates one new view. Numeric options:

`MinVal`	Default value: `0`
`MaxVal`	Default value: `0`
`TimeStep`	Default value: `0`
`Visible`	Default value: `1`
`Dimension`	Default value: `-1`
`View`	Default value: `-1`

`Plugin(FieldFromAmplitudePhase)`

Plugin(FieldFromAmplitudePhase) builds a complex field 'u' from amplitude 'a' (complex) and phase 'phi' given in two different 'Views' u = a * exp(k*phi), with k the wavenumber.

The result is to be interpolated in a sufficiently fine mesh: 'MeshFile'.

Plugin(FieldFromAmplitudePhase) generates one new view. String options:

`MeshFile` Default value: `"fine.msh"`

Numeric options:

`Wavenumber`
Default value: `5`

`AmplitudeView`
Default value: `0`

`PhaseView`
Default value: `1`

`Plugin(Gradient)`

Plugin(Gradient) computes the gradient of the field in the view 'View'.

If 'View' < 0, the plugin is run on the current view.

Plugin(Gradient) creates one new view. Numeric options:

`View` Default value: `-1`

`Plugin(HarmonicToTime)`

Plugin(HarmonicToTime) takes the values in the time steps 'RealPart' and 'ImaginaryPart' of the view 'View', and creates a new view containing

'View'['RealPart'] * cos(p) - 'View'['ImaginaryPart'] * sin(p)

with p = 2*Pi*k/'NumSteps', k = 0, ..., 'NumSteps'-1.

If 'View' < 0, the plugin is run on the current view.

Plugin(HarmonicToTime) creates one new view. Numeric options:

`RealPart` Default value: `0`

`ImaginaryPart`
Default value: `1`

`NumSteps` Default value: `20`

`View` Default value: `-1`

`Plugin(HomologyComputation)`

Plugin(HomologyComputation) computes representative chains of basis elements of (relative) homology and cohomology spaces.

Define physical groups in order to specify the computation domain and the relative subdomain. Otherwise the whole mesh is the domain and the relative subdomain is empty.

Plugin(HomologyComputation) creates new views, one for each basis element. The resulting basis chains of desired dimension together with the mesh are saved to the given file. String options:

`DomainPhysicalGroups`
Default value: `""`

`SubdomainPhysicalGroups`
Default value: `""`

`ReductionImmunePhysicalGroups`
Default value: `""`

`DimensionOfChainsToSave`
Default value: `"0, 1, 2, 3"`

`Filename` Default value: `"homology.msh"`

Numeric options:

`ComputeHomology`
Default value: `1`

`ComputeCohomology`
Default value: `0`

`HomologyPhysicalGroupsBegin`
Default value: `-1`

`CohomologyPhysicalGroupsBegin`
Default value: `-1`

`CreatePostProcessingViews`
Default value: `1`

`ReductionOmit`
Default value: `1`

`ReductionCombine`
Default value: `3`

`PostProcessSimplify`
Default value: `1`

`ReductionHeuristic`
Default value: `1`

`Plugin(HomologyPostProcessing)`
Plugin(HomologyPostProcessing) operates on representative basis chains of homology and cohomology spaces. Functionality:

1. (co)homology basis transformation: 'TransformationMatrix': Integer matrix of the transformation. 'PhysicalGroupsOfOperatedChains': (Co)chains of a (co)homology space basis to be transformed. Results a new (co)chain basis that is an integer cobination of the given basis.

2. Make basis representations of a homology space and a cohomology space compatible: 'PhysicalGroupsOfOperatedChains': Chains of a homology space basis. 'PhysicalGroupsOfOperatedChains2': Cochains of a cohomology space basis. Results a new basis for the homology space such that the incidence

matrix of the new basis and the basis of the cohomology space is the identity matrix.

Options: 'PhysicalGroupsToTraceResults': Trace the resulting (co)chains to the given physical groups. 'PhysicalGroupsToProjectResults': Project the resulting (co)chains to the complement of the given physical groups. 'NameForResultChains': Post-processing view name prefix for the results. 'ApplyBoundaryOperatorToResults': Apply boundary operator to the resulting chains.

String options:

`TransformationMatrix`
Default value: `"1, 0; 0, 1"`

`PhysicalGroupsOfOperatedChains`
Default value: `"1, 2"`

`PhysicalGroupsOfOperatedChains2`
Default value: `""`

`PhysicalGroupsToTraceResults`
Default value: `""`

`PhysicalGroupsToProjectResults`
Default value: `""`

`NameForResultChains`
Default value: `"c"`

Numeric options:

`ApplyBoundaryOperatorToResults`
Default value: `0`

`Plugin(Integrate)`

Plugin(Integrate) integrates a scalar field over all the elements of the view 'View' (if 'Dimension' < 0), or over all elements of the prescribed dimension (if 'Dimension' > 0). If the field is a vector field,the circulation/flux of the field over line/surface elements is calculated.

If 'View' < 0, the plugin is run on the current view.

If 'OverTime' = 1 , the plugin integrates the scalar view over time instead of over space.

Plugin(Integrate) creates one new view. Numeric options:

`View` Default value: `-1`

`OverTime` Default value: `-1`

`Dimension`
Default value: `-1`

`Plugin(Isosurface)`

Plugin(Isosurface) extracts the isosurface of value 'Value' from the view 'View', and draws the 'OtherTimeStep'-th step of the view 'OtherView' on this isosurface.

If 'ExtractVolume' is nonzero, the plugin extracts the isovolume with values greater (if 'ExtractVolume' > 0) or smaller (if 'ExtractVolume' < 0) than the isosurface 'Value'.

If 'OtherTimeStep' < 0, the plugin uses, for each time step in 'View', the corresponding time step in 'OtherView'. If 'OtherView' < 0, the plugin uses 'View' as the value source.

If 'View' < 0, the plugin is run on the current view.

Plugin(Isosurface) creates as many views as there are time steps in 'View'. Numeric options:

`Value` Default value: `0`

`ExtractVolume`
Default value: `0`

`RecurLevel`
Default value: `4`

`TargetError`
Default value: `0`

`View` Default value: `-1`

`OtherTimeStep`
Default value: `-1`

`OtherView`
Default value: `-1`

`Plugin(Lambda2)`

Plugin(Lambda2) computes the eigenvalues Lambda(1,2,3) of the tensor (S_ik S_kj + Om_ik Om_kj), where S_ij = 0.5 (ui,j + uj,i) and Om_ij = 0.5 (ui,j - uj,i) are respectively the symmetric and antisymmetric parts of the velocity gradient tensor.

Vortices are well represented by regions where Lambda(2) is negative.

If 'View' contains tensor elements, the plugin directly uses the tensors as the values of the velocity gradient tensor; if 'View' contains vector elements, the plugin uses them as the velocities from which to derive the velocity gradient tensor.

If 'View' < 0, the plugin is run on the current view.

Plugin(Lambda2) creates one new view. Numeric options:

`Eigenvalue`
Default value: `2`

`View` Default value: `-1`

`Plugin(LongitudeLatitude)`

Plugin(LongituteLatitude) projects the view 'View' in longitude-latitude.

If 'View' < 0, the plugin is run on the current view.

Plugin(LongituteLatitude) is executed in place. Numeric options:

`View` Default value: `-1`

`Plugin(MakeSimplex)`

Plugin(MakeSimplex) decomposes all non-simplectic elements (quadrangles, prisms, hexahedra, pyramids) in the view 'View' into simplices (triangles, tetrahedra).

If 'View' < 0, the plugin is run on the current view.

Plugin(MakeSimplex) is executed in-place. Numeric options:

`View` Default value: `-1`

`Plugin(MathEval)`

Plugin(MathEval) creates a new view using data from the time step 'TimeStep' in the view 'View'.

If only 'Expression0' is given (and 'Expression1', ..., 'Expression8' are all empty), the plugin creates a scalar view. If 'Expression0', 'Expression1' and/or 'Expression2' are given (and 'Expression3', ..., 'Expression8' are all empty) the plugin creates a vector view. Otherwise the plugin creates a tensor view.

In addition to the usual mathematical functions (Exp, Log, Sqrt, Sin, Cos, Fabs, etc.) and operators (+, -, *, /, ^), all expressions can contain:

- the symbols v0, v1, v2, ..., vn, which represent the n components in 'View';

- the symbols w0, w1, w2, ..., wn, which represent the n components of 'OtherView', at time step 'OtherTimeStep';

- the symbols x, y and z, which represent the three spatial coordinates.

If 'TimeStep' < 0, the plugin extracts data from all the time steps in the view.

If 'View' < 0, the plugin is run on the current view.

Plugin(MathEval) creates one new view.If 'PhysicalRegion' < 0, the plugin is run on all physical regions.

Plugin(MathEval) creates one new view. String options:

`Expression0`
Default value: `"Sqrt(v0^2+v1^2+v2^2)"`

`Expression1`
Default value: `""`

`Expression2`
Default value: `""`

`Expression3`
Default value: `""`

`Expression4`
Default value: `""`

`Expression5`
Default value: `""`

`Expression6`
Default value: `""`

`Expression7`
Default value: `""`

`Expression8`
Default value: `""`

Numeric options:

`TimeStep`	Default value: `-1`
`View`	Default value: `-1`
`OtherTimeStep`	Default value: `-1`
`OtherView`	Default value: `-1`
`ForceInterpolation`	Default value: `0`
`PhysicalRegion`	Default value: `-1`

`Plugin(MinMax)`

Plugin(MinMax) computes the min/max of a view.

If 'View' `<` 0, the plugin is run on the current view.

If 'OverTime' = 1, calculates the min/max over space AND time

If 'Argument' = 1, calculates the min/max AND the argmin/argmax

Plugin(MinMax) creates two new views. Numeric options:

`View`	Default value: `-1`
`OverTime`	Default value: `0`
`Argument`	Default value: `0`

`Plugin(ModifyComponent)`

Plugin(ModifyComponent) sets the 'Component'-th component of the 'TimeStep'-th time step in the view 'View' to the expression 'Expression'.

'Expression' can contain:

- the usual mathematical functions (Log, Sqrt, Sin, Cos, Fabs, ...) and operators (+, -, *, /, ^);

- the symbols x, y and z, to retrieve the coordinates of the current node;

- the symbols Time and TimeStep, to retrieve the current time and time step values;

- the symbol v, to retrieve the 'Component'-th component of the field in 'View' at the 'TimeStep'-th time step;

- the symbols v0, v1, v2, ..., v8, to retrieve each component of the field in 'View' at the 'TimeStep'-th time step;

- the symbol w, to retrieve the 'Component'-th component of the field in 'OtherView' at the 'OtherTimeStep'-th time step. If 'OtherView' and 'View' are based on different spatial grids, or if their data types are different, 'OtherView' is interpolated onto 'View';

- the symbols w0, w1, w2, ..., w8, to retrieve each component of the field in 'OtherView' at the 'OtherTimeStep'-th time step.

If 'TimeStep' < 0, the plugin automatically loops over all the time steps in 'View' and evaluates 'Expression' for each one.

If 'OtherTimeStep' < 0, the plugin uses 'TimeStep' instead.

If 'Component' < 0, the plugin automatically ops over all the components in the view and evaluates 'Expression' for each one.

If 'View' < 0, the plugin is run on the current view.

If 'OtherView' < 0, the plugin uses 'View' instead.

Plugin(ModifyComponent) is executed in-place. String options:

`Expression`
Default value: `"v0 * Sin(x)"`

Numeric options:

`Component`
Default value: `-1`

`TimeStep` Default value: `-1`

`View` Default value: `-1`

`OtherTimeStep`
Default value: `-1`

`OtherView`
Default value: `-1`

`ForceInterpolation`
Default value: `0`

`Plugin(ModulusPhase)`

Plugin(ModulusPhase) interprets the time steps 'realPart' and 'imaginaryPart' in the view 'View' as the real and imaginary parts of a complex field and replaces them with their corresponding modulus and phase.

If 'View' < 0, the plugin is run on the current view.

Plugin(ModulusPhase) is executed in-place. Numeric options:

`RealPart` Default value: `0`

`ImaginaryPart`
Default value: `1`

`View` Default value: `-1`

`Plugin(NearToFarField)`

Plugin(NearToFarField) computes the far field pattern from the near electric E and magnetic H fields on a surface enclosing the radiating device (antenna).

Parameters: the wavenumber, the angular discretisation (phi in [0, 2*Pi] and theta in [0, Pi]) of the far field sphere and the indices of the views containing the complex-valued E and H fields. If 'Normalize' is set, the far field is normalized to 1. If 'dB' is set, the far field is computed in dB. If 'NegativeTime' is set, E and H are assumed to have exp(-iwt) time dependency; otherwise they are assume to have exp(+iwt) time dependency. If 'MatlabOutputFile' is given the raw far field data is also exported in Matlab format.

Plugin(NearToFarField) creates one new view. String options:

`MatlabOutputFile`	Default value: `"farfield.m"`

Numeric options:

`Wavenumber`	Default value: `1`
`PhiStart`	Default value: `0`
`PhiEnd`	Default value: `6.28319`
`NumPointsPhi`	Default value: `60`
`ThetaStart`	Default value: `0`
`ThetaEnd`	Default value: `3.14159`
`NumPointsTheta`	Default value: `30`
`EView`	Default value: `0`
`HView`	Default value: `1`
`Normalize`	Default value: `1`
`dB`	Default value: `1`
`NegativeTime`	Default value: `0`

`Plugin(NearestNeighbor)`

Plugin(NearestNeighbor) computes the distance from each point in 'View' to its nearest neighbor.

If 'View' < 0, the plugin is run on the current view.

Plugin(NearestNeighbor) is executed in-place. Numeric options:

`View`	Default value: `-1`

`Plugin(NewView)`

Plugin(NewView) creates a new view from a mesh. Numeric options:

`View`	Default value: `-1`

`Plugin(Particles)`

Plugin(Particles) computes the trajectory of particules in the force field given by the ‘TimeStep’-th time step of a vector view ‘View’.

The plugin takes as input a grid defined by the 3 points (‘X0’,‘Y0’,‘Z0’) (origin), (‘X1’,‘Y1’,‘Z1’) (axis of U) and (‘X2’,‘Y2’,‘Z2’) (axis of V).

The number of particles along U and V that are to be transported is set with the options ‘NumPointsU’ and ‘NumPointsV’. The equation

A2 * d^2X(t)/dt^2 + A1 * dX(t)/dt + A0 * X(t) = F

is then solved with the initial conditions X(t=0) chosen as the grid, dX/dt(t=0)=0, and with F interpolated from the vector view.

Time stepping is done using a Newmark scheme with step size ‘DT’ and ‘MaxIter’ maximum number of iterations.

If ‘View’ < 0, the plugin is run on the current view.

Plugin(Particles) creates one new view containing multi-step vector points. Numeric options:

`X0`	Default value: `0`
`Y0`	Default value: `0`
`Z0`	Default value: `0`
`X1`	Default value: `1`
`Y1`	Default value: `0`
`Z1`	Default value: `0`
`X2`	Default value: `0`
`Y2`	Default value: `1`
`Z2`	Default value: `0`
`NumPointsU`	Default value: `10`
`NumPointsV`	Default value: `1`
`A2`	Default value: `1`
`A1`	Default value: `0`
`A0`	Default value: `0`
`DT`	Default value: `0.1`
`MaxIter`	Default value: `100`
`TimeStep`	Default value: `0`
`View`	Default value: `-1`

`Plugin(Probe)`

Plugin(Probe) gets the value of the view 'View' at the point ('X','Y','Z').

If 'View' < 0, the plugin is run on the current view.

Plugin(Probe) creates one new view. Numeric options:

`X`	Default value: `0`
`Y`	Default value: `0`
`Z`	Default value: `0`
`View`	Default value: `-1`

`Plugin(Remove)`

Plugin(Remove) removes the marked items from the view 'View'.

If 'View' < 0, the plugin is run on the current view.

Plugin(Remove) is executed in-place. Numeric options:

`Text2D`	Default value: `1`
`Text3D`	Default value: `1`
`Points`	Default value: `0`
`Lines`	Default value: `0`
`Triangles`	Default value: `0`
`Quadrangles`	Default value: `0`
`Tetrahedra`	Default value: `0`
`Hexahedra`	Default value: `0`
`Prisms`	Default value: `0`
`Pyramids`	Default value: `0`
`Scalar`	Default value: `1`
`Vector`	Default value: `1`
`Tensor`	Default value: `1`
`View`	Default value: `-1`

`Plugin(Scal2Vec)`

Plugin(Scal2Vec) converts the scalar fields of 'ViewX', 'ViewY' and/or 'ViewZ' into a vectorial field. The new view 'NameNewView' contains it.

If the value of 'ViewX', 'ViewY' or 'ViewZ' is -1, the value of the vectorial field in the corresponding direction is 0. String options:

`NameNewView`	Default value: `"NewView"`

Numeric options:

`ViewX` Default value: `-1`

`ViewY` Default value: `-1`

`ViewZ` Default value: `-1`

`Plugin(SimplePartition)`
Plugin(SimplePartition) partitions the current mesh into 'NumSlices' slices, along the X-, Y- or Z-axis depending on the value of 'Direction' (0,1,2). The plugin creates partition boundaries if 'CreateBoundaries' is set. String options:

`Mapping` Default value: `"t"`

Numeric options:

`NumSlices`
Default value: `4`

`Direction`
Default value: `0`

`CreateBoundaries`
Default value: `1`

`Plugin(Skin)`
Plugin(Skin) extracts the boundary (skin) of the view 'View'. If 'Visible' is set, the plugin only extracts the skin of visible entities.

If 'View' `<` 0, the plugin is run on the current view.

Plugin(Skin) creates one new view. Numeric options:

`Visible` Default value: `1`

`View` Default value: `-1`

`Plugin(Smooth)`
Plugin(Smooth) averages the values at the nodes of the view 'View'.

If 'View' `<` 0, the plugin is run on the current view.

Plugin(Smooth) is executed in-place. Numeric options:

`View` Default value: `-1`

`Plugin(SphericalRaise)`
Plugin(SphericalRaise) transforms the coordinates of the elements in the view 'View' using the values associated with the 'TimeStep'-th time step.

Instead of elevating the nodes along the X, Y and Z axes as with the View['View'].RaiseX, View['View'].RaiseY and View['View'].RaiseZ options, the raise is applied along the radius of a sphere centered at ('Xc', 'Yc', 'Zc').

To produce a standard radiation pattern, set 'Offset' to minus the radius of the sphere the original data lives on.

If 'View' `<` 0, the plugin is run on the current view.

Plugin(SphericalRaise) is executed in-place. Numeric options:

`Xc`	Default value: `0`
`Yc`	Default value: `0`
`Zc`	Default value: `0`
`Raise`	Default value: `1`
`Offset`	Default value: `0`
`TimeStep`	Default value: `0`
`View`	Default value: `-1`

`Plugin(StreamLines)`

Plugin(StreamLines) computes stream lines from the 'TimeStep'-th time step of a vector view 'View' and optionally interpolates the scalar view 'OtherView' on the resulting stream lines.

The plugin takes as input a grid defined by the 3 points ('X0','Y0','Z0') (origin), ('X1','Y1','Z1') (axis of U) and ('X2','Y2','Z2') (axis of V).

The number of points along U and V that are to be transported is set with the options 'NumPointsU' and 'NumPointsV'. The equation

dX(t)/dt = V(x,y,z)

is then solved with the initial condition X(t=0) chosen as the grid and with V(x,y,z) interpolated on the vector view.

The time stepping scheme is a RK44 with step size 'DT' and 'MaxIter' maximum number of iterations.

If 'TimeStep' < 0, the plugin tries to compute streamlines of the unsteady flow.

If 'View' < 0, the plugin is run on the current view.

Plugin(StreamLines) creates one new view. This view contains multi-step vector points if 'OtherView' < 0, or single-step scalar lines if 'OtherView' >= 0. Numeric options:

`X0`	Default value: `0`
`Y0`	Default value: `0`
`Z0`	Default value: `0`
`X1`	Default value: `1`
`Y1`	Default value: `0`
`Z1`	Default value: `0`
`X2`	Default value: `0`
`Y2`	Default value: `1`
`Z2`	Default value: `0`
`NumPointsU`	Default value: `10`
`NumPointsV`	Default value: `1`

`DT`	Default value: `0.1`
`MaxIter`	Default value: `100`
`TimeStep`	Default value: `0`
`View`	Default value: `-1`
`OtherView`	Default value: `-1`

`Plugin(Tetrahedralize)`

Plugin(Tetrahedralize) tetrahedralizes the points in the view 'View'.

If 'View' < 0, the plugin is run on the current view.

Plugin(Tetrahedralize) creates one new view. Numeric options:

`View`	Default value: `-1`

`Plugin(Transform)`

Plugin(Transform) transforms the homogeneous node coordinates (x,y,z,1) of the elements in the view 'View' by the matrix

['A11' 'A12' 'A13' 'Tx'] ['A21' 'A22' 'A23' 'Ty'] ['A31' 'A32' 'A33' 'Tz'].

If 'SwapOrientation' is set, the orientation of the elements is reversed.

If 'View' < 0, the plugin is run on the current view.

Plugin(Transform) is executed in-place. Numeric options:

`A11`	Default value: `1`
`A12`	Default value: `0`
`A13`	Default value: `0`
`A21`	Default value: `0`
`A22`	Default value: `1`
`A23`	Default value: `0`
`A31`	Default value: `0`
`A32`	Default value: `0`
`A33`	Default value: `1`
`Tx`	Default value: `0`
`Ty`	Default value: `0`
`Tz`	Default value: `0`
`SwapOrientation`	Default value: `0`
`View`	Default value: `-1`

`Plugin(Triangulate)`

Plugin(Triangulate) triangulates the points in the view 'View', assuming that all the points belong to a surface that can be projected one-to-one onto a plane.

If 'View' < 0, the plugin is run on the current view.

Plugin(Triangulate) creates one new view. Numeric options:

`View` Default value: `-1`

`Plugin(Warp)`

Plugin(Warp) transforms the elements in the view 'View' by adding to their node coordinates the vector field stored in the 'TimeStep'-th time step of the view 'OtherView', scaled by 'Factor'.

If 'View' < 0, the plugin is run on the current view.

If 'OtherView' < 0, the vector field is taken as the field of surface normals multiplied by the 'TimeStep' value in 'View'. (The smoothing of the surface normals is controlled by the 'SmoothingAngle' parameter.)

Plugin(Warp) is executed in-place. Numeric options:

`Factor` Default value: `1`

`TimeStep` Default value: `0`

`SmoothingAngle`
Default value: `180`

`View` Default value: `-1`

`OtherView`
Default value: `-1`

8.3 Post-processing options

General post-processing option names have the form '`PostProcessing.`*`string`*'. Options peculiar to post-processing views take two forms.

1. options that should apply to all views can be set through '`View.`*`string`*', *before any view is loaded*;
2. options that should apply only to the n-th view take the form '`View[`*`n`*`].`*`string`*' ($n = 0, 1, 2, \ldots$), *after the n-th view is loaded.*

The list of all post-processing and view options is given in Section B.5 [Post-processing options list], page 182. See Section A.8 [t8.geo], page 118, and Section A.9 [t9.geo], page 121, for some examples.

9 File formats

This chapter describes Gmsh's native "MSH" file format, used to store meshes and associated post-processing datasets. The MSH format exists in two flavors: ASCII and binary. The format has a version number (currently: 2.2) that is independent of Gmsh's main version number.

(Remember that for small post-processing datasets you can also use human-readable "parsed" post-processing views, as described in Section 8.1 [Post-processing commands], page 62. Such "parsed" views do not require an underlying mesh, and can therefore be easier to use in some cases.)

9.1 MSH ASCII file format

The MSH ASCII file format contains one mandatory section giving information about the file (`$MeshFormat`), followed by several optional sections defining the nodes (`$Nodes`), elements (`$Elements`), region names (`$PhysicalName`), periodicity relations (`$Periodic`) and post-processing datasets (`$NodeData`, `$ElementData`, `$ElementNodeData`).

When `$Elements` are given, `$Nodes` should also be provided, before the `$Elements` section. Currently only one `$Nodes` and one `$Elements` section are allowed per file. (This might/will change in the future.)

Any section with an unrecognized header is simply ignored: you can thus add comments in a '`.msh`' file by putting them e.g. inside a `$Comments`/`$EndComments` section.

Sections can be repeated in the same file, and post-processing sections can be put into separate files (e.g. one file per time step). Nodes are assumed to be defined before elements.

The format is defined as follows:

```
$MeshFormat
version-number file-type data-size
$EndMeshFormat
$Nodes
number-of-nodes
node-number x-coord y-coord z-coord
...
$EndNodes
$Elements
number-of-elements
elm-number elm-type number-of-tags < tag > ... node-number-list
...
$EndElements
$Periodic
number-of-periodic-entities
dimension slave-entity-tag master-entity-tag
number-of-nodes
slave-node-number master-node-number
...
$EndPeriodic
$PhysicalNames
```

```
number-of-names
physical-dimension physical-number "physical-name"
...
$EndPhysicalNames
$NodeData
number-of-string-tags
< "string-tag" >
...
number-of-real-tags
< real-tag >
...
number-of-integer-tags
< integer-tag >
...
node-number value ...
...
$EndNodeData
$ElementData
number-of-string-tags
< "string-tag" >
...
number-of-real-tags
< real-tag >
...
number-of-integer-tags
< integer-tag >
...
elm-number value ...
...
$EndElementData
$ElementNodeData
number-of-string-tags
< "string-tag" >
...
number-of-real-tags
< real-tag >
...
number-of-integer-tags
< integer-tag >
...
elm-number number-of-nodes-per-element value ...
...
$EndElementNodeData
$InterpolationScheme
"name"
number-of-element-topologies
elm-topology
```

```
number-of-interpolation-matrices
num-rows num-columns value ...
...
$EndInterpolationScheme
```

where

version-number
: is a real number equal to 2.2

file-type
: is an integer equal to 0 in the ASCII file format.

data-size
: is an integer equal to the size of the floating point numbers used in the file (currently only *data-size* = sizeof(double) is supported).

number-of-nodes
: is the number of nodes in the mesh.

node-number
: is the number (index) of the *n*-th node in the mesh; *node-number* must be a postive (non-zero) integer. Note that the *node-number*s do not necessarily have to form a dense nor an ordered sequence.

x-coord y-coord z-coord
: are the floating point values giving the X, Y and Z coordinates of the *n*-th node.

number-of-elements
: is the number of elements in the mesh.

elm-number
: is the number (index) of the *n*-th element in the mesh; *elm-number* must be a postive (non-zero) integer. Note that the *elm-number*s do not necessarily have to form a dense nor an ordered sequence.

elm-type
: defines the geometrical type of the *n*-th element:

`1`	2-node line.
`2`	3-node triangle.
`3`	4-node quadrangle.
`4`	4-node tetrahedron.
`5`	8-node hexahedron.
`6`	6-node prism.
`7`	5-node pyramid.
`8`	3-node second order line (2 nodes associated with the vertices and 1 with the edge).
`9`	6-node second order triangle (3 nodes associated with the vertices and 3 with the edges).

`10`	9-node second order quadrangle (4 nodes associated with the vertices, 4 with the edges and 1 with the face).
`11`	10-node second order tetrahedron (4 nodes associated with the vertices and 6 with the edges).
`12`	27-node second order hexahedron (8 nodes associated with the vertices, 12 with the edges, 6 with the faces and 1 with the volume).
`13`	18-node second order prism (6 nodes associated with the vertices, 9 with the edges and 3 with the quadrangular faces).
`14`	14-node second order pyramid (5 nodes associated with the vertices, 8 with the edges and 1 with the quadrangular face).
`15`	1-node point.
`16`	8-node second order quadrangle (4 nodes associated with the vertices and 4 with the edges).
`17`	20-node second order hexahedron (8 nodes associated with the vertices and 12 with the edges).
`18`	15-node second order prism (6 nodes associated with the vertices and 9 with the edges).
`19`	13-node second order pyramid (5 nodes associated with the vertices and 8 with the edges).
`20`	9-node third order incomplete triangle (3 nodes associated with the vertices, 6 with the edges)
`21`	10-node third order triangle (3 nodes associated with the vertices, 6 with the edges, 1 with the face)
`22`	12-node fourth order incomplete triangle (3 nodes associated with the vertices, 9 with the edges)
`23`	15-node fourth order triangle (3 nodes associated with the vertices, 9 with the edges, 3 with the face)
`24`	15-node fifth order incomplete triangle (3 nodes associated with the vertices, 12 with the edges)
`25`	21-node fifth order complete triangle (3 nodes associated with the vertices, 12 with the edges, 6 with the face)
`26`	4-node third order edge (2 nodes associated with the vertices, 2 internal to the edge)
`27`	5-node fourth order edge (2 nodes associated with the vertices, 3 internal to the edge)
`28`	6-node fifth order edge (2 nodes associated with the vertices, 4 internal to the edge)
`29`	20-node third order tetrahedron (4 nodes associated with the vertices, 12 with the edges, 4 with the faces)

`30`	35-node fourth order tetrahedron (4 nodes associated with the vertices, 18 with the edges, 12 with the faces, 1 in the volume)
`31`	56-node fifth order tetrahedron (4 nodes associated with the vertices, 24 with the edges, 24 with the faces, 4 in the volume)
`92`	64-node third order hexahedron (8 nodes associated with the vertices, 24 with the edges, 24 with the faces, 8 in the volume)
`93`	125-node fourth order hexahedron (8 nodes associated with the vertices, 36 with the edges, 54 with the faces, 27 in the volume)

See below for the ordering of the nodes.

`number-of-tags`
: gives the number of integer tags that follow for the *n*-th element. By default, the first *tag* is the number of the physical entity to which the element belongs; the second is the number of the elementary geometrical entity to which the element belongs; the third is the number of mesh partitions to which the element belongs, followed by the partition ids (negative partition ids indicate ghost cells). A zero tag is equivalent to no tag.

`node-number-list`
: is the list of the node numbers of the *n*-th element. The ordering of the nodes is given in Section 9.3 [Node ordering], page 94.

`number-of-string-tags`
: gives the number of string tags that follow. By default the first *string-tag* is interpreted as the name of the post-processing view and the second as the name of the interpolation scheme. The interpolation scheme is provided in the `$InterpolationScheme` section (see below).

`number-of-real-tags`
: gives the number of real number tags that follow. By default the first *real-tag* is interpreted as a time value associated with the dataset.

`number-of-integer-tags`
: gives the number of integer tags that follow. By default the first *integer-tag* is interpreted as a time step index (starting at 0), the second as the number of field components of the data in the view (1, 3 or 9), the third as the number of entities (nodes or elements) in the view, and the fourth as the partition index for the view data (0 for no partition).

`number-of-nodes-per-elements`
: gives the number of node values for an element in an element-based view.

`value`
: is a real number giving the value associated with a node or an element. For `NodeData` (respectively `ElementData`) views, there are *ncomp* values per node (resp. per element), where *ncomp* is the number of field components. For `ElementNodeData` views, there are *ncomp* times *number-of-nodes-per-elements* values per element.

`number-of-element-topologies`
: is the number of element topologies for which interpolation matrices are provided

elm-topology
: is the id tag of a given element topology: 1 for points, 2 for lines, 3 for triangles, 4 for quadrangles, 5 for tetrahedra, 6 for pyramids, 7 for prisms, 8 for hexahedra, 9 for polygons and 10 for polyhedra.

number-of-interpolation-matrices
: is the number of interpolation matrices provided for this element topology. Currently you should provide 2 matrices, i.e., the matrices that specify how to interpolate the data (they have the same meaning as in Section 8.1 [Post-processing commands], page 62). The matrices are specified by 2 integers (*num-rows* and *num-columns*) followed by the values.

Below is a small example (a mesh consisting of two quadrangles with an associated nodal scalar dataset; the comments are not part of the actual file!):

```
$MeshFormat
2.2 0 8
$EndMeshFormat
$Nodes
6                       six mesh nodes:
1 0.0 0.0 0.0             node #1: coordinates (0.0, 0.0, 0.0)
2 1.0 0.0 0.0             node #2: coordinates (1.0, 0.0, 0.0)
3 1.0 1.0 0.0             etc.
4 0.0 1.0 0.0
5 2.0 0.0 0.0
6 2.0 1.0 0.0
$EndNodes
$Elements
2                       two elements:
1 3 2 99 2 1 2 3 4        quad #1: type 3, physical 99, elementary 2, nodes 1 2 3 4
2 3 2 99 2 2 5 6 3        quad #2: type 3, physical 99, elementary 2, nodes 2 5 6 3
$EndElements
$NodeData
1                       one string tag:
"A scalar view"           the name of the view ("A scalar view")
1                       one real tag:
0.0                       the time value (0.0)
3                       three integer tags:
0                         the time step (0; time steps always start at 0)
1                         1-component (scalar) field
6                         six associated nodal values
1 0.0                   value associated with node #1 (0.0)
2 0.1                   value associated with node #2 (0.1)
3 0.2                   etc.
4 0.0
5 0.2
6 0.4
$EndNodeData
```

9.2 MSH binary file format

The binary file format is similar to the ASCII format described above:

```
$MeshFormat
version-number file-type data-size
one-binary
$EndMeshFormat
```

```
$Nodes
number-of-nodes
nodes-binary
$EndNodes
$Elements
number-of-elements
element-header-binary
elements-binary
element-header-binary
elements-binary
...
$EndElements

[ all other sections are identical to ASCII, except that node-number,
  elm-number, number-of-nodes-per-element and values are written in
  binary format ]
```

where

version-number
: is a real number equal to 2.2.

file-type
: is an integer equal to 1.

data-size
: has the same meaning as in the ASCII file format. Currently only *data-size* = sizeof(double) is supported.

one-binary
: is an integer of value 1 written in binary form. This integer is used for detecting if the computer on which the binary file was written and the computer on which the file is read are of the same type (little or big endian).

  Here is a pseudo C code to write *one-binary*:

```
int one = 1;
fwrite(&one, sizeof(int), 1, file);
```

number-of-nodes
: has the same meaning as in the ASCII file format.

nodes-binary
: is the list of nodes in binary form, i.e., a array of *number-of-nodes* * (4 + 3 * *data-size*) bytes. For each node, the first 4 bytes contain the node number and the next (3 * *data-size*) bytes contain the three floating point coordinates.

  Here is a pseudo C code to write *nodes-binary*:

```
for(i = 0; i < number_of_nodes; i++){
  fwrite(&num_i, sizeof(int), 1, file);
  double xyz[3] = {node_i_x, node_i_y, node_i_z};
  fwrite(&xyz, sizeof(double), 3, file);
}
```

`number-of-elements`
: has the same meaning as in the ASCII file format.

`element-header-binary`
: is a list of 3 integers in binary form, i.e., an array of (3 * 4) bytes: the first four bytes contain the type of the elements that follow (same as *elm-type* in the ASCII format), the next four contain the number of elements that follow, and the last four contain the number of tags per element (same as *number-of-tags* in the ASCII format).

Here is a pseudo C code to write *element-header-binary*:

```
int header[3] = {elm_type, num_elm_follow, num_tags};
fwrite(&header, sizeof(int), 2, file);
```

`elements-binary`
: is a list of elements in binary form, i.e., an array of "number of elements that follow" * (4 + *number-of-tags* * 4 + *#node-number-list* * 4) bytes. For each element, the first four bytes contain the element number, the next (*number-of-tags* * 4) contain the tags, and the last (*#node-number-list* * 4) contain the node indices.

Here is a pseudo C code to write *elements-binary* for triangles with the 2 standard tags (the physical and elementary regions):

```
for(i = 0; i < number_of_triangles; i++){
  int data[6] = {num_i, physical, elementary,
                 node_i_1, node_i_2, node_i_3};
  fwrite(data, sizeof(int), 6, file);
}
```

9.3 Node ordering

Historically, Gmsh developpers have started by implementing linear elements (lines, triangles, quads, tets, prisms and hexes). Then, second and sometimes third order elements have been hardcoded. We here distinguish "low order elements" that have been hardcoded initially and "high order elements" that have been coded in a general fashion, theoretically valid for any order.

9.3.1 Low order elements

For all mesh and post-processing file formats, the reference elements are defined as follows.

```
Line:                 Line3:          Line4:

0----------1 --> u    0-----2----1    0----2----3----1
```

```
Triangle:               Triangle6:          Triangle9/10:          Triangle12/15:

v
^                                                                   2
|                                                                   | \
2                       2                    2                      9   8
|‘\                     |‘\                  | \                    |     \
|  ‘\                   |  ‘\                7   6                 10 (14)  7
|    ‘\                 5    ‘4              |     \                |         \
|      ‘\               |      ‘\            8  (9)  5             11 (12) (13) 6
|        ‘\             |        ‘\          |         \            |             \
0----------1 --> u      0-----3----1         0---3---4---1          0---3---4---5---1



Quadrangle:            Quadrangle8:            Quadrangle9:

      v
      ^
      |
3-----------2          3-----6-----2           3-----6-----2
|     |     |          |           |           |           |
|     |     |          |           |           |           |
|     +---- | --> u    7           5           7     8     5
|           |          |           |           |           |
|           |          |           |           |           |
0-----------1          0-----4-----1           0-----4-----1



Tetrahedron:                          Tetrahedron10:

                   v
                 .
               ,/
              /
           2                                     2
         ,/|‘\                                 ,/|‘\
       ,/  |  ‘\                             ,/  |  ‘\
     ,/    ’.   ‘\                         ,6    ’.   ‘5
   ,/       |     ‘\                     ,/       8     ‘\
 ,/         |       ‘\                 ,/         |       ‘\
0-----------’.--------1 --> u         0--------4--’.--------1
 ‘\.         |      ,/                 ‘\.         |      ,/
    ‘\.      |    ,/                      ‘\.      |    ,9
       ‘\.   ’. ,/                           ‘7.   ’. ,/
          ‘\. |/                                ‘\. |/
             ‘3                                    ‘3
                ‘\.
                   ‘ w
```

```
Hexahedron:             Hexahedron20:          Hexahedron27:

       v
3----------2            3----13----2           3----13----2
|\     ^   |\           |\         |\          |\         |\
| \    |   | \          | 15       | 14        |15    24  | 14
|  \   |   |  \         9  \       11 \        9  \ 20    11 \
|   7------+---6        |   7----19+---6       |   7----19+---6
|   |  +-- |-- | -> u   |   |      |   |       |22 |  26  | 23|
0---+---\--1   |        0---+-8----1   |       0---+-8----1   |
 \  |    \  \  |         \  17      \  18       \ 17    25 \  18
  \ |     \  \ |         10 |        12|        10 |  21    12|
   \|      w  \|           \|         \|          \|         \|
    4----------5            4----16----5           4----16----5



Prism:                      Prism15:               Prism18:

           w
           ^
           |
           3                       3                      3
         ,/|`\                   ,/|`\                  ,/|`\
       ,/  |  `\               12  |  13              12  |  13
     ,/    |    `\           ,/    |    `\          ,/    |    `\
    4------+------5         4------14-----5         4------14-----5
    |      |      |         |      8      |         |      8      |
    |    ,/|`\    |         |      |      |         |    ,/|`\    |
    |  ,/  |  `\  |         |      |      |         |  15  |  16  |
    |,/    |    `\|         |      |      |         |,/    |    `\|
   ,|      |      |\        10     |      11        10-----17-----11
 ,/ |      0      | `\      |      0      |         |      0      |
u   |    ,/ `\    |    v    |    ,/ `\    |         |    ,/ `\    |
    |  ,/     `\  |         |  ,6     `7  |         |  ,6     `7  |
    |,/         `\|         |,/         `\|         |,/         `\|
    1-------------2         1------9------2         1------9------2



Pyramid:                     Pyramid13:                   Pyramid14:

               4                            4                            4
             ,/|\                         ,/|\                         ,/|\
           ,/ .'|\                      ,/ .'|\                      ,/ .'|\
         ,/   | | \                   ,/   | | \                   ,/   | | \
       ,/    .' | `.                ,/    .' | `.                ,/    .' | `.
     ,/      |  '.  \             ,7      |  12  \             ,7      |  12  \
   ,/       .' w |   \          ,/       .'   |   \          ,/       .'   |   \
 ,/         |  ^ |    \       ,/         9    |    11      ,/         9    |    11
0----------.'--|-3    `.     0--------6-.'----3    `.     0--------6-.'----3    `.
 `\        |   |  `\    \      `\        |      `\    \     `\        |      `\    \
   `\     .'   +----`\ - \ -> v  `5     .'        10   \      `5     .' 13     10   \
     `\   |    `\     `\  \        `\   |           `\  \       `\   |           `\  \
       `\.'      `\     `\`          `\.'             `\`         `\.'             `\`
          1----------------2            1--------8-------2           1--------8-------2
                    `\
                       u
```

9.3.2 High order elements

The node ordering of a higher order (possibly curved) element is compatible with the numbering of low order element (it is a generalization). We number nodes in the following order:

- the element principal or corner vertices;
- the internal nodes for each edge;
- the internal nodes for each face;
- the volume internal nodes.

The numbering for internal nodes is recursive, ie. the numbering follows that of the nodes of an embedded edge/face/volume of lower order. The higher order nodes are assumed to be equispaced. Edges and faces are numbered following the lowest order template that generates a single high-order on this edge/face. Furthermore, an edge is oriented from the vertex with the lowest to the highest index. The orientation of a face is such that the computed normal points outward; the starting point is the vertex with the lowest index.

9.4 Legacy formats

This section describes Gmsh's older native file formats. Future versions of Gmsh will continue to support these formats, but we recommend that you do not use them in new applications.

9.4.1 MSH file format version 1.0 (Legacy)

The MSH file format version 1.0 is Gmsh's old native mesh file format, now superseded by the format described in Section 9.1 [MSH ASCII file format], page 87. It is defined as follows:

```
$NOD
number-of-nodes
node-number x-coord y-coord z-coord
...
$ENDNOD
$ELM
number-of-elements
elm-number elm-type reg-phys reg-elem number-of-nodes node-number-list
...
$ENDELM
```

where

number-of-nodes
: is the number of nodes in the mesh.

node-number
: is the number (index) of the *n*-th node in the mesh; *node-number* must be a postive (non-zero) integer. Note that the *node-number*s do not necessarily have to form a dense nor an ordered sequence.

x-coord y-coord z-coord
: are the floating point values giving the X, Y and Z coordinates of the *n*-th node.

`number-of-elements`
: is the number of elements in the mesh.

`elm-number`
: is the number (index) of the *n*-th element in the mesh; *elm-number* must be a postive (non-zero) integer. Note that the *elm-number*s do not necessarily have to form a dense nor an ordered sequence.

`elm-type`
: defines the geometrical type of the *n*-th element:

`1`	2-node line.
`2`	3-node triangle.
`3`	4-node quadrangle.
`4`	4-node tetrahedron.
`5`	8-node hexahedron.
`6`	6-node prism.
`7`	5-node pyramid.
`8`	3-node second order line (2 nodes associated with the vertices and 1 with the edge).
`9`	6-node second order triangle (3 nodes associated with the vertices and 3 with the edges).
`10`	9-node second order quadrangle (4 nodes associated with the vertices, 4 with the edges and 1 with the face).
`11`	10-node second order tetrahedron (4 nodes associated with the vertices and 6 with the edges).
`12`	27-node second order hexahedron (8 nodes associated with the vertices, 12 with the edges, 6 with the faces and 1 with the volume).
`13`	18-node second order prism (6 nodes associated with the vertices, 9 with the edges and 3 with the quadrangular faces).
`14`	14-node second order pyramid (5 nodes associated with the vertices, 8 with the edges and 1 with the quadrangular face).
`15`	1-node point.
`16`	8-node second order quadrangle (4 nodes associated with the vertices and 4 with the edges).
`17`	20-node second order hexahedron (8 nodes associated with the vertices and 12 with the edges).
`18`	15-node second order prism (6 nodes associated with the vertices and 9 with the edges).
`19`	13-node second order pyramid (5 nodes associated with the vertices and 8 with the edges).

See below for the ordering of the nodes.

`reg-phys` is the number of the physical entity to which the element belongs; *reg-phys* must be a postive integer, or zero. If *reg-phys* is equal to zero, the element is considered not to belong to any physical entity.

`reg-elem` is the number of the elementary entity to which the element belongs; *reg-elem* must be a postive (non-zero) integer.

`number-of-nodes`
is the number of nodes for the *n*-th element. This is redundant, but kept for backward compatibility.

`node-number-list`
is the list of the *number-of-nodes* node numbers of the *n*-th element. The ordering of the nodes is given in Section 9.3 [Node ordering], page 94.

9.4.2 POS ASCII file format (Legacy)

The POS ASCII file is Gmsh's old native post-processing format, now superseded by the format described in Section 9.1 [MSH ASCII file format], page 87. It is defined as follows:

```
$PostFormat
1.4 file-type data-size
$EndPostFormat
$View
view-name nb-time-steps
nb-scalar-points nb-vector-points nb-tensor-points
nb-scalar-lines nb-vector-lines nb-tensor-lines
nb-scalar-triangles nb-vector-triangles nb-tensor-triangles
nb-scalar-quadrangles nb-vector-quadrangles nb-tensor-quadrangles
nb-scalar-tetrahedra nb-vector-tetrahedra nb-tensor-tetrahedra
nb-scalar-hexahedra nb-vector-hexahedra nb-tensor-hexahedra
nb-scalar-prisms nb-vector-prisms nb-tensor-prisms
nb-scalar-pyramids nb-vector-pyramids nb-tensor-pyramids
nb-scalar-lines2 nb-vector-lines2 nb-tensor-lines2
nb-scalar-triangles2 nb-vector-triangles2 nb-tensor-triangles2
nb-scalar-quadrangles2 nb-vector-quadrangles2 nb-tensor-quadrangles2
nb-scalar-tetrahedra2 nb-vector-tetrahedra2 nb-tensor-tetrahedra2
nb-scalar-hexahedra2 nb-vector-hexahedra2 nb-tensor-hexahedra2
nb-scalar-prisms2 nb-vector-prisms2 nb-tensor-prisms2
nb-scalar-pyramids2 nb-vector-pyramids2 nb-tensor-pyramids2
nb-text2d nb-text2d-chars nb-text3d nb-text3d-chars
time-step-values
< scalar-point-value > ... < vector-point-value > ...
    < tensor-point-value > ...
< scalar-line-value > ... < vector-line-value > ...
    < tensor-line-value > ...
< scalar-triangle-value > ... < vector-triangle-value > ...
    < tensor-triangle-value > ...
< scalar-quadrangle-value > ... < vector-quadrangle-value > ...
```

```
    < tensor-quadrangle-value > ...
< scalar-tetrahedron-value > ... < vector-tetrahedron-value > ...
    < tensor-tetrahedron-value > ...
< scalar-hexahedron-value > ... < vector-hexahedron-value > ...
    < tensor-hexahedron-value > ...
< scalar-prism-value > ... < vector-prism-value > ...
    < tensor-prism-value > ...
< scalar-pyramid-value > ... < vector-pyramid-value > ...
    < tensor-pyramid-value > ...
< scalar-line2-value > ... < vector-line2-value > ...
    < tensor-line2-value > ...
< scalar-triangle2-value > ... < vector-triangle2-value > ...
    < tensor-triangle2-value > ...
< scalar-quadrangle2-value > ... < vector-quadrangle2-value > ...
    < tensor-quadrangle2-value > ...
< scalar-tetrahedron2-value > ... < vector-tetrahedron2-value > ...
    < tensor-tetrahedron2-value > ...
< scalar-hexahedron2-value > ... < vector-hexahedron2-value > ...
    < tensor-hexahedron2-value > ...
< scalar-prism2-value > ... < vector-prism2-value > ...
    < tensor-prism2-value > ...
< scalar-pyramid2-value > ... < vector-pyramid2-value > ...
    < tensor-pyramid2-value > ...
< text2d > ... < text2d-chars > ...
< text3d > ... < text3d-chars > ...
$EndView
```

where

file-type
: is an integer equal to 0 in the ASCII file format.

data-size
: is an integer equal to the size of the floating point numbers used in the file (usually, *data-size* = sizeof(double)).

view-name
: is a string containing the name of the view (max. 256 characters).

nb-time-steps
: is an integer giving the number of time steps in the view.

nb-scalar-points
nb-vector-points
...
: are integers giving the number of scalar points, vector points, ..., in the view.

nb-text2d
nb-text3d
: are integers giving the number of 2D and 3D text strings in the view.

`nb-text2d-chars`
`nb-text3d-chars`
: are integers giving the total number of characters in the 2D and 3D strings.

`time-step-values`
: is a list of *nb-time-steps* double precision numbers giving the value of the time (or any other variable) for which an evolution was saved.

`scalar-point-value`
`vector-point-value`
`...`
: are lists of double precision numbers giving the node coordinates and the values associated with the nodes of the *nb-scalar-points* scalar points, *nb-vector-points* vector points, . . ., for each of the *time-step-values*.

  For example, *vector-triangle-value* is defined as:

```
coord1-node1 coord1-node2 coord1-node3
coord2-node1 coord2-node2 coord2-node3
coord3-node1 coord3-node2 coord3-node3
comp1-node1-time1 comp2-node1-time1 comp3-node1-time1
comp1-node2-time1 comp2-node2-time1 comp3-node2-time1
comp1-node3-time1 comp2-node3-time1 comp3-node3-time1
comp1-node1-time2 comp2-node1-time2 comp3-node1-time2
comp1-node2-time2 comp2-node2-time2 comp3-node2-time2
comp1-node3-time2 comp2-node3-time2 comp3-node3-time2
...
```

  The ordering of the nodes is given in Section 9.3 [Node ordering], page 94.

`text2d`
: is a list of 4 double precision numbers:

```
coord1 coord2 style index
```

  where *coord1* and *coord2* give the X-Y position of the 2D string in screen coordinates (measured from the top-left corner of the window) and where *index* gives the starting index of the string in *text2d-chars*. If *coord1* (respectively *coord2*) is negative, the position is measured from the right (respectively bottom) edge of the window. If *coord1* (respectively *coord2*) is larger than 99999, the string is centered horizontally (respectively vertically). If *style* is equal to zero, the text is aligned bottom-left and displayed using the default font and size. Otherwise, *style* is converted into an integer whose eight lower bits give the font size, whose eight next bits select the font (the index corresponds to the position in the font menu in the GUI), and whose eight next bits define the text alignment (0=bottom-left, 1=bottom-center, 2=bottom-right, 3=top-left, 4=top-center, 5=top-right, 6=center-left, 7=center-center, 8=center-right).

`text2d-chars`
: is a list of *nb-text2d-chars* characters. Substrings are separated with the null '`\0`' character.

`text3d`
: is a list of 5 double precision numbers

```
coord1 coord2 coord3 style index
```

where *coord1*, *coord2* and *coord3* give the XYZ coordinates of the string in model (real world) coordinates, *index* gives the starting index of the string in *text3d-chars*, and *style* has the same meaning as in *text2d*.

`text3d-chars`
is a list of *nb-text3d-chars* chars. Substrings are separated with the null '`\0`' character.

9.4.3 POS binary file format (Legacy)

The POS binary file format is the same as the POS ASCII file format described in Section 9.4.2 [POS ASCII file format], page 99, except that:

1. *file-type* equals 1.
2. all lists of floating point numbers and characters are written in binary format
3. there is an additional integer, of value 1, written before *time-step-values*. This integer is used for detecting if the computer on which the binary file was written and the computer on which the file is read are of the same type (little or big endian).

Here is a pseudo C code to write a post-processing file in binary format:

```
int one = 1;

fprintf(file, "$PostFormat\n");
fprintf(file, "%g %d %d\n", 1.4, 1, sizeof(double));
fprintf(file, "$EndPostFormat\n");
fprintf(file, "$View\n");
fprintf(file, "%s %d "
  "%d %d %d %d %d %d %d %d %d "
  "%d %d %d %d %d %d %d %d %d "
  "%d %d %d %d %d %d %d %d %d "
  "%d %d %d %d %d %d %d %d %d "
  "%d %d %d %d %d %d %d %d %d "
  "%d %d %d %d\n",
  view-name, nb-time-steps,
  nb-scalar-points, nb-vector-points, nb-tensor-points,
  nb-scalar-lines, nb-vector-lines, nb-tensor-lines,
  nb-scalar-triangles, nb-vector-triangles, nb-tensor-triangles,
  nb-scalar-quadrangles, nb-vector-quadrangles, nb-tensor-quadrangles,
  nb-scalar-tetrahedra, nb-vector-tetrahedra, nb-tensor-tetrahedra,
  nb-scalar-hexahedra, nb-vector-hexahedra, nb-tensor-hexahedra,
  nb-scalar-prisms, nb-vector-prisms, nb-tensor-prisms,
  nb-scalar-pyramids, nb-vector-pyramids, nb-tensor-pyramids,
  nb-scalar-lines2, nb-vector-lines2, nb-tensor-lines2,
  nb-scalar-triangles2, nb-vector-triangles2, nb-tensor-triangles2,
  nb-scalar-quadrangles2, nb-vector-quadrangles2, nb-tensor-quadrangles2,
  nb-scalar-tetrahedra2, nb-vector-tetrahedra2, nb-tensor-tetrahedra2,
  nb-scalar-hexahedra2, nb-vector-hexahedra2, nb-tensor-hexahedra2,
  nb-scalar-prisms2, nb-vector-prisms2, nb-tensor-prisms2,
  nb-scalar-pyramids2, nb-vector-pyramids2, nb-tensor-pyramids2,
  nb-text2d, nb-text2d-chars, nb-text3d, nb-text3d-chars);
fwrite(&one, sizeof(int), 1, file);
fwrite(time-step-values, sizeof(double), nb-time-steps, file);
fwrite(all-scalar-point-values, sizeof(double), ..., file);
...
fprintf(file, "\n$EndView\n");
```

In this pseudo-code, *all-scalar-point-values* is the array of double precision numbers containing all the *scalar-point-value* lists, put one after each other in order to form a long array of doubles. The principle is the same for all other kinds of values.

Appendix A Tutorial

The following examples introduce new features gradually, starting with ‘t1.geo’. The files corresponding to these examples are available in the ‘tutorial’ directory of the Gmsh distribution.

To learn how to run Gmsh on your computer, see Chapter 3 [Running Gmsh on your system], page 11. Screencasts that show how to use the GUI are available on http://geuz.org/gmsh/screencasts/.

A.1 ‘t1.geo’

```
/*********************************************************************
 *
 *  Gmsh tutorial 1
 *
 *  Variables, elementary entities (points, lines, surfaces), physical
 *  entities (points, lines, surfaces)
 *
 *********************************************************************/

// The simplest construction in Gmsh's scripting language is the
// `affectation'. The following command defines a new variable `lc':

lc = 1e-2;

// This variable can then be used in the definition of Gmsh's simplest
// `elementary entity', a `Point'. A Point is defined by a list of
// four numbers: three coordinates (X, Y and Z), and a characteristic
// length (lc) that sets the target element size at the point:

Point(1) = {0, 0, 0, lc};

// The distribution of the mesh element sizes is then obtained by
// interpolation of these characteristic lengths throughout the
// geometry. Another method to specify characteristic lengths is to
// use a background mesh (see `t7.geo' and `bgmesh.pos').

// We can then define some additional points as well as our first
// curve.  Curves are Gmsh's second type of elementery entities, and,
// amongst curves, straight lines are the simplest. A straight line is
// defined by a list of point numbers. In the commands below, for
// example, the line 1 starts at point 1 and ends at point 2:

Point(2) = {.1, 0,  0, lc} ;
Point(3) = {.1, .3, 0, lc} ;
Point(4) = {0,  .3, 0, lc} ;
```

```
Line(1) = {1,2} ;
Line(2) = {3,2} ;
Line(3) = {3,4} ;
Line(4) = {4,1} ;

// The third elementary entity is the surface. In order to define a
// simple rectangular surface from the four lines defined above, a
// line loop has first to be defined. A line loop is a list of
// connected lines, a sign being associated with each line (depending
// on the orientation of the line):

Line Loop(5) = {4,1,-2,3} ;

// We can then define the surface as a list of line loops (only one
// here, since there are no holes--see ‘t4.geo’):

Plane Surface(6) = {5} ;

// At this level, Gmsh knows everything to display the rectangular
// surface 6 and to mesh it. An optional step is needed if we want to
// associate specific region numbers to the various elements in the
// mesh (e.g. to the line segments discretizing lines 1 to 4 or to the
// triangles discretizing surface 6). This is achieved by the
// definition of ‘physical entities’. Physical entities will group
// elements belonging to several elementary entities by giving them a
// common number (a region number).

// We can for example group the points 1 and 2 into the physical
// entity 1:

Physical Point(1) = {1,2} ;

// Consequently, two punctual elements will be saved in the output
// mesh file, both with the region number 1. The mechanism is
// identical for line or surface elements:

MyLine = 99;
Physical Line(MyLine) = {1,2,4} ;

Physical Surface("My fancy surface label") = {6} ;

// All the line elements created during the meshing of lines 1, 2 and
// 4 will be saved in the output mesh file with the region number 99;
// and all the triangular elements resulting from the discretization
// of surface 6 will be given an automatic region number (100,
// associated with the label "My fancy surface label").
// Note that if no physical entities are defined, then all the
// elements in the mesh will be saved "as is", with their default
// orientation.
```

A.2 ‘t2.geo’

```
/*********************************************************************
 *
 *  Gmsh tutorial 2
 *
 *  Includes, geometrical transformations, extruded geometries,
 *  elementary entities (volumes), physical entities (volumes)
 *
 *********************************************************************/

// We first include the previous tutorial file, in order to use it as
// a basis for this one:

Include "t1.geo";

// We can then add new points and lines in the same way as we did in
// ‘t1.geo’:

Point(5) = {0, .4, 0, lc};
Line(5) = {4, 5};

// But Gmsh also provides tools to tranform (translate, rotate, etc.)
// elementary entities or copies of elementary entities. For example,
// the point 3 can be moved by 0.05 units to the left with:

Translate {-0.05, 0, 0} { Point{3}; }

// The resulting point can also be duplicated and translated by 0.1
// along the y axis:

Translate {0, 0.1, 0} { Duplicata{ Point{3}; } }

// This command created a new point with an automatically assigned
// id. This id can be obtained using the graphical user interface by
// hovering the mouse over it and looking at the bottom of the graphic
// window: in this case, the new point has id "6". Point 6 can then be
// used to create new entities, e.g.:

Line(7) = {3, 6};
Line(8) = {6, 5};
Line Loop(10) = {5,-8,-7,3};
Plane Surface(11) = {10};
```

```
// Using the graphical user interface to obtain the ids of newly
// created entities can sometimes be cumbersome. It can then be
// advantageous to use the return value of the transformation commands
// directly. For example, the Translate command returns a list
// containing the ids of the translated entities. For example, we can
// translate copies of the two surfaces 6 and 11 to the right with the
// following command:

my_new_surfs[] = Translate {0.12, 0, 0} { Duplicata{ Surface{6, 11}; } };

// my_new_surfs[] (note the square brackets) denotes a list, which in
// this case contains the ids of the two new surfaces (check
// 'Tools->Message console' to see the message):

Printf("New surfaces '%g' and '%g'", my_new_surfs[0], my_new_surfs[1]);

// In Gmsh lists use square brackets for their definition (mylist[] =
// {1,2,3};) as well as to access their elements (myotherlist[] =
// {mylist[0], mylist[2]};). Note that list indexing starts at 0.

// Volumes are the fourth type of elementary entities in Gmsh. In the
// same way one defines line loops to build surfaces, one has to
// define surface loops (i.e. 'shells') to build volumes. The
// following volume does not have holes and thus consists of a single
// surface loop:

Point(100) = {0., 0.3, 0.13, lc};  Point(101) = {0.08, 0.3, 0.1, lc};
Point(102) = {0.08, 0.4, 0.1, lc}; Point(103) = {0., 0.4, 0.13, lc};

Line(110) = {4, 100};   Line(111) = {3, 101};
Line(112) = {6, 102};   Line(113) = {5, 103};
Line(114) = {103, 100}; Line(115) = {100, 101};
Line(116) = {101, 102}; Line(117) = {102, 103};

Line Loop(118) = {115, -111, 3, 110};  Plane Surface(119) = {118};
Line Loop(120) = {111, 116, -112, -7}; Plane Surface(121) = {120};
Line Loop(122) = {112, 117, -113, -8}; Plane Surface(123) = {122};
Line Loop(124) = {114, -110, 5, 113};  Plane Surface(125) = {124};
Line Loop(126) = {115, 116, 117, 114}; Plane Surface(127) = {126};

Surface Loop(128) = {127, 119, 121, 123, 125, 11};
Volume(129) = {128};

// When a volume can be extruded from a surface, it is usually easier
// to use the Extrude command directly instead of creating all the
// points, lines and surfaces by hand. For example, the following
// command extrudes the surface 11 along the z axis and automatically
// creates a new volume (as well as all the needed points, lines and
// surfaces):

Extrude {0, 0, 0.12} { Surface{my_new_surfs[1]}; }

// The following command permits to manually assign a characteristic
// length to some of the new points:

Characteristic Length {103, 105, 109, 102, 28, 24, 6, 5} = lc * 3;

// Note that, if the transformation tools are handy to create complex
// geometries, it is also sometimes useful to generate the ‘flat’
// geometry, with an explicit list of all elementary entities. This
// can be achieved by selecting the ‘File->Save as->Gmsh unrolled
// geometry’ menu or by typing
//
// > gmsh t2.geo -0
//
// on the command line.

// To save all the tetrahedra discretizing the volumes 129 and 130
// with a common region number, we finally define a physical
// volume:

Physical Volume (1) = {129,130};
```

A.3 ‘t3.geo’

```
/*********************************************************************
 *
 *  Gmsh tutorial 3
 *
 *  Extruded meshes, parameters, options
 *
 *********************************************************************/

// Again, we start by including the first tutorial:

Include "t1.geo";

// As in ‘t2.geo’, we plan to perform an extrusion along the z axis.
// But here, instead of only extruding the geometry, we also want to
// extrude the 2D mesh. This is done with the same ‘Extrude’ command,
// but by specifying element ’Layers’ (2 layers in this case, the
// first one with 8 subdivisions and the second one with 2
// subdivisions, both with a height of h/2):
```

```
h = 0.1;

Extrude {0,0,h} {
  Surface{6}; Layers{ {8,2}, {0.5,1} };
}

// The extrusion can also be performed with a rotation instead of a
// translation, and the resulting mesh can be recombined into prisms
// (we use only one layer here, with 7 subdivisions). All rotations
// are specified by an axis direction ({0,1,0}), an axis point
// ({-0.1,0,0.1}) and a rotation angle (-Pi/2):

Extrude { {0,1,0} , {-0.1,0,0.1} , -Pi/2 } {
  Surface{122}; Layers{7}; Recombine;
}

// Note that a translation ({-2*h,0,0}) and a rotation ({1,0,0},
// {0,0.15,0.25}, Pi/2) can also be combined. Here the angle is
// specified as a 'parameter', using the 'DefineConstant' syntax.
// This parameter can be modified insteractively in the GUI, and
// can be exchanged with other codes using the ONELAB framework:

DefineConstant[ angle = {90, Min 0, Max 120, Step 1,
                         Label "Twisting angle", Path "Parameters"} ];

out[] = Extrude { {-2*h,0,0}, {1,0,0} , {0,0.15,0.25} , angle * Pi / 180 } {
  Surface{144}; Layers{10}; Recombine;
};

// In this last extrusion command we retrieved the volume number
// programatically by using the return value (a list) of the Extrude
// command. This list contains the "top" of the extruded surface (in
// out[0]), the newly created volume (in out[1]) and the ids of the
// lateral surfaces (in out[2], out[3], ...)

// We can then define a new physical volume to save all the tetrahedra
// with a common region number (101):

Physical Volume(101) = {1, 2, out[1]};

// Let us now change some options... Since all interactive options are
// accessible in Gmsh's scripting language, we can for example define
// a global characteristic length factor or redefine some colors
// directly in the input file:

Mesh.CharacteristicLengthFactor = 4;
```

```
General.Color.Text = White;
Geometry.Color.Points = Orange;
Mesh.Color.Points = {255,0,0};

// Note that all colors can be defined literally or numerically, i.e.
// ‘Mesh.Color.Points = Red’ is equivalent to ‘Mesh.Color.Points =
// {255,0,0}’; and also note that, as with user-defined variables, the
// options can be used either as right or left hand sides, so that the
// following command will set the surface color to the same color as
// the points:

Geometry.Color.Surfaces = Geometry.Color.Points;

// You can use the ‘Help->Current options’ menu to see the current
// values of all options. To save all the options in a file, use
// ‘File->Save as->Gmsh options’. To associate the current options
// with the current file use ‘File->Save Options->For Current
// File’. To save the current options for all future Gmsh sessions use
// ‘File->Save Options->As default’.
```

A.4 ‘t4.geo’

```
/*********************************************************************
 *
 *  Gmsh tutorial 4
 *
 *  Built-in functions, holes, strings, mesh color
 *
 *********************************************************************/

// As usual, we start by defining some variables:

cm = 1e-02;
e1 = 4.5 * cm; e2 = 6 * cm / 2; e3 =  5 * cm / 2;
h1 = 5 * cm; h2 = 10 * cm; h3 = 5 * cm; h4 = 2 * cm; h5 = 4.5 * cm;
R1 = 1 * cm; R2 = 1.5 * cm; r = 1 * cm;
Lc1 = 0.01;
Lc2 = 0.003;

// We can use all the usual mathematical functions (note the
// capitalized first letters), plus some useful functions like
// Hypot(a, b) := Sqrt(a^2 + b^2):

ccos = (-h5*R1 + e2 * Hypot(h5, Hypot(e2, R1))) / (h5^2 + e2^2);
ssin = Sqrt(1 - ccos^2);

// Then we define some points and some lines using these variables:
```

```
Point(1) = {-e1-e2, 0    , 0, Lc1}; Point(2) = {-e1-e2, h1   , 0, Lc1};
Point(3) = {-e3-r , h1   , 0, Lc2}; Point(4) = {-e3-r , h1+r , 0, Lc2};
Point(5) = {-e3   , h1+r , 0, Lc2}; Point(6) = {-e3   , h1+h2, 0, Lc1};
Point(7) = { e3   , h1+h2, 0, Lc1}; Point(8) = { e3   , h1+r , 0, Lc2};
Point(9) = { e3+r , h1+r , 0, Lc2}; Point(10)= { e3+r , h1   , 0, Lc2};
Point(11)= { e1+e2, h1   , 0, Lc1}; Point(12)= { e1+e2, 0    , 0, Lc1};
Point(13)= { e2   , 0    , 0, Lc1};

Point(14)= { R1 / ssin, h5+R1*ccos, 0, Lc2};
Point(15)= { 0        , h5        , 0, Lc2};
Point(16)= {-R1 / ssin, h5+R1*ccos, 0, Lc2};
Point(17)= {-e2       , 0.0       , 0, Lc1};

Point(18)= {-R2 , h1+h3   , 0, Lc2}; Point(19)= {-R2 , h1+h3+h4, 0, Lc2};
Point(20)= { 0  , h1+h3+h4, 0, Lc2}; Point(21)= { R2 , h1+h3+h4, 0, Lc2};
Point(22)= { R2 , h1+h3   , 0, Lc2}; Point(23)= { 0  , h1+h3   , 0, Lc2};

Point(24)= { 0, h1+h3+h4+R2, 0, Lc2}; Point(25)= { 0, h1+h3-R2,    0, Lc2};

Line(1)  = {1 , 17};
Line(2)  = {17, 16};

// Gmsh provides other curve primitives than stright lines: splines,
// B-splines, circle arcs, ellipse arcs, etc. Here we define a new
// circle arc, starting at point 14 and ending at point 16, with the
// circle’s center being the point 15:

Circle(3) = {14,15,16};

// Note that, in Gmsh, circle arcs should always be smaller than
// Pi. We can then define additional lines and circles, as well as a
// new surface:

Line(4)  = {14,13}; Line(5)   = {13,12};  Line(6)  = {12,11};
Line(7)  = {11,10}; Circle(8) = {8,9,10}; Line(9)  = {8,7};
Line(10) = {7,6};   Line(11)  = {6,5};    Circle(12) = {3,4,5};
Line(13) = {3,2};   Line(14)  = {2,1};    Line(15) = {18,19};
Circle(16) = {21,20,24}; Circle(17) = {24,20,19};
Circle(18) = {18,23,25}; Circle(19) = {25,23,22};
Line(20) = {21,22};

Line Loop(21) = {17,-15,18,19,-20,16};
Plane Surface(22) = {21};

// But we still need to define the exterior surface. Since this
// surface has a hole, its definition now requires two lines loops:
```

```
Line Loop(23) = {11,-12,13,14,1,2,-3,4,5,6,7,-8,9,10};
Plane Surface(24) = {23,21};

// As a general rule, if a surface has N holes, it is defined by N+1
// line loops: the first loop defines the exterior boundary; the other
// loops define the boundaries of the holes.

// Finally, we can add some comments by embedding a post-processing
// view containing some strings, and change the color of some mesh
// entities:

View "comments" {
  // Add a text string in window coordinates, 10 pixels from the left
  // and 10 pixels from the bottom:
  T2(10, -10, 0){ "Copyright (C) My Company" };

  // Add another text string in window coordinates, 10 pixels from the
  // left and 15 pixels from the top, using the StrCat() function to
  // concatenate a string with the current date:
  T2(10, 15, 0){ StrCat("File created on ", Today) };

  // Add a text string in model coordinates at (X,Y,Z) = (0, 0.11, 0):
  T3(0, 0.11, 0, 0){ "Hole" };
};

Color Grey50{ Surface{ 22 }; }
Color Purple{ Surface{ 24 }; }
Color Red{ Line{ 1:14 }; }
Color Yellow{ Line{ 15:20 }; }
```

A.5 ‘t5.geo’

```
/*********************************************************************
 *
 *  Gmsh tutorial 5
 *
 *  Characteristic lengths, arrays of variables, functions, loops
 *
 *********************************************************************/

// We start by defining some target mesh sizes:

lcar1 = .1;
lcar2 = .0005;
lcar3 = .055;
```

```
// If we wanted to change these mesh sizes globally (without changing
// the above definitions), we could give a global scaling factor for
// all characteristic lengths on the command line with the ‘-clscale’
// option (or with ‘Mesh.CharacteristicLengthFactor’ in an option
// file). For example, with:
//
// > gmsh t5.geo -clscale 1
//
// this input file produces a mesh of approximately 1,300 nodes and
// 11,000 tetrahedra. With
//
// > gmsh t5.geo -clscale 0.2
//
// the mesh counts approximately 350,000 nodes and 2.1 million
// tetrahedra. You can check mesh statistics in the graphical user
// interface with the ‘Tools->Statistics’ menu.

// We proceed by defining some elementary entities describing a
// truncated cube:

Point(1) = {0.5,0.5,0.5,lcar2}; Point(2) = {0.5,0.5,0,lcar1};
Point(3) = {0,0.5,0.5,lcar1};   Point(4) = {0,0,0.5,lcar1};
Point(5) = {0.5,0,0.5,lcar1};   Point(6) = {0.5,0,0,lcar1};
Point(7) = {0,0.5,0,lcar1};     Point(8) = {0,1,0,lcar1};
Point(9) = {1,1,0,lcar1};       Point(10) = {0,0,1,lcar1};
Point(11) = {0,1,1,lcar1};      Point(12) = {1,1,1,lcar1};
Point(13) = {1,0,1,lcar1};      Point(14) = {1,0,0,lcar1};

Line(1) = {8,9};    Line(2) = {9,12};  Line(3) = {12,11};
Line(4) = {11,8};   Line(5) = {9,14};  Line(6) = {14,13};
Line(7) = {13,12};  Line(8) = {11,10}; Line(9) = {10,13};
Line(10) = {10,4};  Line(11) = {4,5};  Line(12) = {5,6};
Line(13) = {6,2};   Line(14) = {2,1};  Line(15) = {1,3};
Line(16) = {3,7};   Line(17) = {7,2};  Line(18) = {3,4};
Line(19) = {5,1};   Line(20) = {7,8};  Line(21) = {6,14};

Line Loop(22) = {-11,-19,-15,-18};   Plane Surface(23) = {22};
Line Loop(24) = {16,17,14,15};       Plane Surface(25) = {24};
Line Loop(26) = {-17,20,1,5,-21,13}; Plane Surface(27) = {26};
Line Loop(28) = {-4,-1,-2,-3};       Plane Surface(29) = {28};
Line Loop(30) = {-7,2,-5,-6};        Plane Surface(31) = {30};
Line Loop(32) = {6,-9,10,11,12,21};  Plane Surface(33) = {32};
Line Loop(34) = {7,3,8,9};           Plane Surface(35) = {34};
Line Loop(36) = {-10,18,-16,-20,4,-8}; Plane Surface(37) = {36};
Line Loop(38) = {-14,-13,-12,19};    Plane Surface(39) = {38};

// Instead of using included files, we now use a user-defined function
// in order to carve some holes in the cube:

Function CheeseHole

  // In the following commands we use the reserved variable name
  // ‘newp’, which automatically selects a new point number. This
  // number is chosen as the highest current point number, plus
  // one. (Note that, analogously to ‘newp’, the variables ‘newc’,
  // ‘news’, ‘newv’ and ‘newreg’ select the highest number amongst
  // currently defined curves, surfaces, volumes and ‘any entities
  // other than points’, respectively.)

  p1 = newp; Point(p1) = {x,  y,  z,  lcar3} ;
  p2 = newp; Point(p2) = {x+r,y,  z,  lcar3} ;
  p3 = newp; Point(p3) = {x,  y+r,z,  lcar3} ;
  p4 = newp; Point(p4) = {x,  y,  z+r,lcar3} ;
  p5 = newp; Point(p5) = {x-r,y,  z,  lcar3} ;
  p6 = newp; Point(p6) = {x,  y-r,z,  lcar3} ;
  p7 = newp; Point(p7) = {x,  y,  z-r,lcar3} ;

  c1 = newreg; Circle(c1) = {p2,p1,p7}; c2 = newreg; Circle(c2) = {p7,p1,p5};
  c3 = newreg; Circle(c3) = {p5,p1,p4}; c4 = newreg; Circle(c4) = {p4,p1,p2};
  c5 = newreg; Circle(c5) = {p2,p1,p3}; c6 = newreg; Circle(c6) = {p3,p1,p5};
  c7 = newreg; Circle(c7) = {p5,p1,p6}; c8 = newreg; Circle(c8) = {p6,p1,p2};
  c9 = newreg; Circle(c9) = {p7,p1,p3}; c10 = newreg; Circle(c10) = {p3,p1,p4};
  c11 = newreg; Circle(c11) = {p4,p1,p6}; c12 = newreg; Circle(c12) = {p6,p1,p7};

  // We need non-plane surfaces to define the spherical holes. Here we
  // use ruled surfaces, which can have 3 or 4 sides:

  l1 = newreg; Line Loop(l1) = {c5,c10,c4};   Ruled Surface(newreg) = {l1};
  l2 = newreg; Line Loop(l2) = {c9,-c5,c1};   Ruled Surface(newreg) = {l2};
  l3 = newreg; Line Loop(l3) = {c12,-c8,-c1}; Ruled Surface(newreg) = {l3};
  l4 = newreg; Line Loop(l4) = {c8,-c4,c11};  Ruled Surface(newreg) = {l4};
  l5 = newreg; Line Loop(l5) = {-c10,c6,c3};  Ruled Surface(newreg) = {l5};
  l6 = newreg; Line Loop(l6) = {-c11,-c3,c7}; Ruled Surface(newreg) = {l6};
  l7 = newreg; Line Loop(l7) = {-c2,-c7,-c12};Ruled Surface(newreg) = {l7};
  l8 = newreg; Line Loop(l8) = {-c6,-c9,c2};  Ruled Surface(newreg) = {l8};

  // We then store the surface loops identification numbers in a list
  // for later reference (we will need these to define the final
  // volume):

  theloops[t] = newreg ;

  Surface Loop(theloops[t]) = {l8+1,l5+1,l1+1,l2+1,l3+1,l7+1,l6+1,l4+1};
  thehole = newreg ;
  Volume(thehole) = theloops[t] ;

Return

// We can use a ‘For’ loop to generate five holes in the cube:

x = 0 ; y = 0.75 ; z = 0 ; r = 0.09 ;

For t In {1:5}

  x += 0.166 ;
  z += 0.166 ;

  // We call the ‘CheeseHole’ function:

  Call CheeseHole ;

  // We define a physical volume for each hole:

  Physical Volume (t) = thehole ;

  // We also print some variables on the terminal (note that, since
  // all variables are treated internally as floating point numbers,
  // the format string should only contain valid floating point format
  // specifiers like ‘%g’, ‘%f’, ’%e’, etc.):

  Printf("Hole %g (center = {%g,%g,%g}, radius = %g) has number %g!",
         t, x, y, z, r, thehole) ;

EndFor

// We can then define the surface loop for the exterior surface of the
// cube:

theloops[0] = newreg ;

Surface Loop(theloops[0]) = {35,31,29,37,33,23,39,25,27} ;

// The volume of the cube, without the 5 holes, is now defined by 6
// surface loops: the first surface loop defines the exterior surface;
// the surface loops other than the first one define holes.  (Again,
// to reference an array of variables, its identifier is followed by
// square brackets):

Volume(186) = {theloops[]} ;
```

```
// We finally define a physical volume for the elements discretizing
// the cube, without the holes (whose elements were already tagged
// with numbers 1 to 5 in the ‘For’ loop):

Physical Volume (10) = 186 ;
```

A.6 ‘t6.geo’

```
/*********************************************************************
 *
 *  Gmsh tutorial 6
 *
 *  Transfinite meshes
 *
 *********************************************************************/

// Let’s use the geometry from the first tutorial as a basis for this
// one
Include "t1.geo";

// Delete the left line and create replace it with 3 new ones
Delete{ Surface{6}; Line{4}; }

p1 = newp; Point(p1) = {-0.05, 0.05, 0, lc};
p2 = newp; Point(p2) = {-0.05, 0.1, 0, lc};

l1 = newl; Line(l1) = {1, p1};
l2 = newl; Line(l2) = {p1, p2};
l3 = newl; Line(l3) = {p2, 4};

// Create surface
Line Loop(1) = {2, -1, l1, l2, l3, -3};
Plane Surface(1) = {1};

// Put 20 points with a refinement toward the extremities on curve 2
Transfinite Line{2} = 20 Using Bump 0.05;

// Put 20 points total on combination of curves l1, l2 and l3 (beware
// that the points p1 and p2 are shared by the curves, so we do not
// create 6 + 6 + 10 = 22 points, but 20!)
Transfinite Line{l1} = 6;
Transfinite Line{l2} = 6;
Transfinite Line{l3} = 10;

// Put 30 points following a geometric progression on curve 1
// (reversed) and on curve 3
Transfinite Line{-1,3} = 30 Using Progression 1.2;
```

```
// Define the Surface as transfinite, by specifying the four corners
// of the transfinite interpolation
Transfinite Surface{1} = {1,2,3,4};

// (Note that the list on the right hand side refers to points, not
// curves. When the surface has only 3 or 4 points on its boundary the
// list can be omitted. The way triangles are generated can be
// controlled by appending "Left", "Right" or "Alternate" after the
// list.)

// Recombine the triangles into quads
Recombine Surface{1};

// Apply an elliptic smoother to the grid
Mesh.Smoothing = 100;

Physical Surface(1) = 1;
```

A.7 't7.geo'

```
/*********************************************************************
 *
 *  Gmsh tutorial 7
 *
 *  Background mesh
 *
 *********************************************************************/

// Characteristic lengths can be specified very accuractely by
// providing a background mesh, i.e., a post-processing view that
// contains the target mesh sizes.

// Merge the first tutorial
Merge "t1.geo";

// Merge a post-processing view containing the target mesh sizes
Merge "bgmesh.pos";

// Apply the view as the current background mesh
Background Mesh View[0];
```

A.8 't8.geo'

```
/*********************************************************************
 *
 *  Gmsh tutorial 8
 *
 *  Post-processing, scripting, animations, options
 *
 *********************************************************************/

// We first include ‘t1.geo’ as well as some post-processing views:

Include "t1.geo";
Include "view1.pos";
Include "view1.pos";
Include "view4.pos";

// We then set some general options:

General.Trackball = 0;
General.RotationX = 0; General.RotationY = 0; General.RotationZ = 0;
General.Color.Background = White; General.Color.Foreground = Black;
General.Color.Text = Black;
General.Orthographic = 0;
General.Axes = 0; General.SmallAxes = 0;

// We also set some options for each post-processing view:

v0 = PostProcessing.NbViews-4;
v1 = v0+1; v2 = v0+2; v3 = v0+3;

View[v0].IntervalsType = 2;
View[v0].OffsetZ = 0.05;
View[v0].RaiseZ = 0;
View[v0].Light = 1;
View[v0].ShowScale = 0;
View[v0].SmoothNormals = 1;

View[v1].IntervalsType = 1;
View[v1].ColorTable = { Green, Blue };
View[v1].NbIso = 10;
View[v1].ShowScale = 0;

View[v2].Name = "Test...";
View[v2].Axes = 1;
View[v2].Color.Axes = Black;
View[v2].IntervalsType = 2;
View[v2].Type = 2;
View[v2].IntervalsType = 2;
View[v2].AutoPosition = 0;
View[v2].PositionX = 85;
View[v2].PositionY = 50;
View[v2].Width = 200;
```

```
View[v2].Height = 130;

View[v3].Visible = 0;

// We then loop from 1 to 255 with a step of 1. (To use a different
// step, just add a third argument in the list. For example, ‘For num
// In {0.5:1.5:0.1}’ would increment num from 0.5 to 1.5 with a step
// of 0.1.)

t = 0;

//For num In {1:1}
For num In {1:255}

  View[v0].TimeStep = t;
  View[v1].TimeStep = t;
  View[v2].TimeStep = t;
  View[v3].TimeStep = t;

  t = (View[v0].TimeStep < View[v0].NbTimeStep-1) ? t+1 : 0;

  View[v0].RaiseZ += 0.01/View[v0].Max * t;

  If (num == 3)
    // We want to create 320x240 frames when num == 3:
    General.GraphicsWidth = 320;
    General.GraphicsHeight = 240;
  EndIf

  // It is possible to nest loops:
  For num2 In {1:50}

    General.RotationX += 10;
    General.RotationY = General.RotationX / 3;
    General.RotationZ += 0.1;

    Sleep 0.01; // sleep for 0.01 second
    Draw; // draw the scene

    If (num == 3)
      // The ‘Print’ command saves the graphical window; the ‘Sprintf’
      // function permits to create the file names on the fly:
      Print Sprintf("t8-%02g.gif", num2);
      Print Sprintf("t8-%02g.jpg", num2);
    EndIf

  EndFor
  If(num == 3)
    // Here we could make a system call to generate a movie. For example,

    // with whirlgif:
    //
    // System "whirlgif -minimize -loop -o t8.gif t8-*.gif";

    // with mpeg_encode:
    //
    // System "mpeg_encode t8.par";

    // with mencoder:
    //
    // System "mencoder 'mf://*.jpg' -mf fps=5 -o t8.mpg -ovc lavc
    //         -lavcopts vcodec=mpeg1video:vhq";
    // System "mencoder 'mf://*.jpg' -mf fps=5 -o t8.mpg -ovc lavc
    //         -lavcopts vcodec=mpeg4:vhq";

    // with ffmpeg:
    //
    // System "ffmpeg -hq -r 5 -b 800 -vcodec mpeg1video
    //         -i t8-%02d.jpg t8.mpg"
    // System "ffmpeg -hq -r 5 -b 800 -i t8-%02d.jpg t8.asf"
  EndIf

EndFor
```

A.9 't9.geo'

```
/*********************************************************************
 *
 *  Gmsh tutorial 9
 *
 *  Post-processing plugins (levelsets, sections, annotations)
 *
 *********************************************************************/

// Plugins can be added to Gmsh in order to extend its
// capabilities. For example, post-processing plugins can modify a
// view, or create a new view based on previously loaded
// views. Several default plugins are statically linked with Gmsh,
// e.g. Isosurface, CutPlane, CutSphere, Skin, Transform or Smooth.
// Plugins can be controlled in the same way as other options: either
// from the graphical interface (right click on the view button, then
// 'Plugins'), or from the command file.
// Let us for example include a three-dimensional scalar view:

Include "view3.pos" ;

// We then set some options for the ‘Isosurface’ plugin (which
// extracts an isosurface from a 3D scalar view), and run it:

Plugin(Isosurface).Value = 0.67 ; // iso-value level
Plugin(Isosurface).View = 0 ; // source view is View[0]
Plugin(Isosurface).Run ; // run the plugin!

// We also set some options for the ‘CutPlane’ plugin (which computes
// a section of a 3D view using the plane A*x+B*y+C*z+D=0), and then
// run it:

Plugin(CutPlane).A = 0 ;
Plugin(CutPlane).B = 0.2 ;
Plugin(CutPlane).C = 1 ;
Plugin(CutPlane).D = 0 ;
Plugin(CutPlane).View = 0 ;
Plugin(CutPlane).Run ;

// Add a title (By convention, for window coordinates a value greater
// than 99999 represents the center. We could also use
// ‘General.GraphicsWidth / 2’, but that would only center the string
// for the current window size.):

Plugin(Annotate).Text = "A nice title" ;
Plugin(Annotate).X = 1.e5;
Plugin(Annotate).Y = 50 ;
Plugin(Annotate).Font = "Times-BoldItalic" ;
Plugin(Annotate).FontSize = 28 ;
Plugin(Annotate).Align = "Center" ;
Plugin(Annotate).View = 0 ;
Plugin(Annotate).Run ;

Plugin(Annotate).Text = "(and a small subtitle)" ;
Plugin(Annotate).Y = 70 ;
Plugin(Annotate).Font = "Times-Roman" ;
Plugin(Annotate).FontSize = 12 ;
Plugin(Annotate).Run ;

// We finish by setting some options:

View[0].Light = 1;
View[0].IntervalsType = 1;
View[0].NbIso = 6;
```

```
View[0].SmoothNormals = 1;
View[1].IntervalsType = 2;
View[2].IntervalsType = 2;
```

A.10 't10.geo'

```
/*********************************************************************
 *
 *  Gmsh tutorial 10
 *
 *  General mesh size fields
 *
 *********************************************************************/

// In addition to specifying target mesh sizes at the points of the
// geometry (see t1) or using a background mesh (see t7), you can use
// general mesh size "Fields".

// Let's create a simple rectangular geometry
lc = .15;
Point(1) = {0.0,0.0,0,lc}; Point(2) = {1,0.0,0,lc};
Point(3) = {1,1,0,lc};     Point(4) = {0,1,0,lc};
Point(5) = {0.2,.5,0,lc};

Line(1) = {3,2}; Line(2) = {2,1}; Line(3) = {1,4}; Line(4) = {4,3};

Line Loop(5) = {1,2,3,4}; Plane Surface(6) = {5};

// Say we would like to obtain mesh elements with size lc/30 near line
// 1 and point 5, and size lc elsewhere. To achieve this, we can use
// two fields: "Attractor", and "Threshold". We first define an
// Attractor field (Field[1]) on points 5 and on line 1. This field
// returns the distance to point 5 and to (100 equidistant points on)
// line 1.
Field[1] = Attractor;
Field[1].NodesList = {5};
Field[1].NNodesByEdge = 100;
Field[1].EdgesList = {1};

// We then define a Threshold field, which uses the return value of
// the Attractor Field[1] in order to define a simple change in
// element size around the attractors (i.e., around point 5 and line
// 1)
//
// LcMax -                         /------------------
//                               /
//                             /
//                            /
// LcMin -o----------------/
//        |                |       |
//     Attractor       DistMin   DistMax
Field[2] = Threshold;
Field[2].IField = 1;
Field[2].LcMin = lc / 30;
Field[2].LcMax = lc;
Field[2].DistMin = 0.15;
Field[2].DistMax = 0.5;

// Say we want to modulate the mesh element sizes using a mathematical
// function of the spatial coordinates. We can do this with the
// MathEval field:
Field[3] = MathEval;
Field[3].F = "Cos(4*3.14*x) * Sin(4*3.14*y) / 10 + 0.101";

// We could also combine MathEval with values coming from other
// fields. For example, let's define an Attractor around point 1
Field[4] = Attractor;
Field[4].NodesList = {1};

// We can then create a MathEval field with a function that depends on
// the return value of the Attractr Field[4], i.e., depending on the
// distance to point 1 (here using a cubic law, with minumum element
// size = lc / 100)
Field[5] = MathEval;
Field[5].F = Sprintf("F4^3 + %g", lc / 100);

// We could also use a Box field to impose a step change in element
// sizes inside a box
Field[6] = Box;
Field[6].VIn = lc / 15;
Field[6].VOut = lc;
Field[6].XMin = 0.3;
Field[6].XMax = 0.6;
Field[6].YMin = 0.3;
Field[6].YMax = 0.6;

// Many other types of fields are available: see the reference manual
// for a complete list. You can also create fields directly in the
// graphical user interface by selecting Define->Fields in the Mesh
// module.

// Finally, let's use the minimum of all the fields as the background
// mesh field
Field[7] = Min;
```

```
Field[7].FieldsList = {2, 3, 5, 6};
Background Field = 7;

// Don't extend the elements sizes from the boundary inside the domain
Mesh.CharacteristicLengthExtendFromBoundary = 0;
```

A.11 't11.geo'

```
/*********************************************************************
 *
 *  Gmsh tutorial 11
 *
 *  Unstructured quadrangular meshes
 *
 *********************************************************************/

// We have seen in tutorials t3 and t6 that extruded and transfinite
// meshes can be "recombined" into quads/prisms/hexahedra by using the
// "Recombine" keyword. Unstructured meshes can be recombined in the
// same way. Let's define a simple geometry with an analytical mesh
// size field:

Point(1) = {-1.25, -.5, 0}; Point(2) = {-1.25, 1.25, 0};
Point(3) = {1.25, -.5, 0};  Point(4) = {1.25, 1.25, 0};

Line(1) = {1, 2}; Line(2) = {2, 4};
Line(3) = {4, 3}; Line(4) = {3, 1};

Line Loop(4) = {1,2, 3, 4}; Plane Surface(100) = {4};

Field[1] = MathEval;
Field[1].F = "0.01*(1.0+30.*(y-x*x)*(y-x*x) + (1-x)*(1-x))";
Background Field = 1;

// To generate quadrangles instead of triangles, we can simply add
Recombine Surface{100};

// If we'd had several surfaces, we could have used 'Recombine Surface
// "*";'. Yet another way would be to specify the global option
// "Mesh.RecombineAll = 1;".

// The default recombination algorithm is called "Blossom": it uses a
// minimum cost perfect matching algorithm to generate fully
// quadrilateral meshes from triangulations. More details about the
// algorithm can be found in the following paper: J.-F. Remacle,
// J. Lambrechts, B. Seny, E. Marchandise, A. Johnen and C. Geuzaine,
// "Blossom-Quad: a non-uniform quadrilateral mesh generator using a
// minimum cost perfect matching algorithm", International Journal for
// Numerical Methods in Engineering, 2011 (in press).

// For even better quadrilateral meshes, you can try the experimental
// "Delaunay for quads" (DelQuad) meshing algorithm: DelQuad is a
// triangulation algorithm that enables to create right triangles
// almost everywhere. Uncomment the following line to try DelQuad:

Mesh.Algorithm = 8; // DelQuad (experimental)
```

A.12 ‘t12.geo’

```
/*********************************************************************
 *
 *  Gmsh tutorial 12
 *
 *  Cross-patch meshing with compounds
 *
 *********************************************************************/

// Compound geometrical entities can be defined to compute a new
// parametrization of groups of elementary geometrical entities. This
// parametrization can then be used for remeshing the compound as if
// it were a single CAD entity.

lc = 0.2;

Point(1) = {0, 0, 0, lc};       Point(2) = {1, 0, 0, lc};
Point(3) = {1, 1, 0.5, lc};     Point(4) = {0, 1, 0.4, lc};
Point(5) = {0.3, 0.2, 0, lc};   Point(6) = {0, 0.01, 0.01, lc};
Point(7) = {0, 0.02, 0.02, lc}; Point(8) = {1, 0.05, 0.02, lc};
Point(9) = {1, 0.32, 0.02, lc};

Line(1) = {1, 2}; Line(2) = {2, 8}; Line(3) = {8, 9};
Line(4) = {9, 3}; Line(5) = {3, 4}; Line(6) = {4, 7};
Line(7) = {7, 6}; Line(8) = {6, 1}; Spline(9) = {7, 5, 9};
Line(10) = {6, 8};

Line Loop(11) = {5, 6, 9, 4};     Ruled Surface(12) = {11};
Line Loop(13) = {9, -3, -10, -7}; Ruled Surface(14) = {13};
Line Loop(15) = {10, -2, -1, -8}; Ruled Surface(16) = {15};

// Treat lines 2, 3 and 4 as a single line
Compound Line(100) = {2, 3, 4};
// Idem with lines 6, 7 and 8
Compound Line(101) = {6, 7, 8};
```

```
// Treat surfaces 12, 14 and 16 as a single surface
Compound Surface(200) = {12, 14, 16};

// Hide the original surfaces so we only see the compound
// (cross-patch) mesh
//Hide {Surface{12, 14, 16}; }

// More details about the reparametrization technique can be found in
// the following papers:
//
// * J.-F. Remacle, C. Geuzaine, G. Compre and E. Marchandise,
//   "High-Quality Surface Remeshing Using Harmonic Maps",
//   International Journal for Numerical Methods in Engineering,
//   83 (4), pp. 403-425, 2010.
//
// * E. Marchandise, G. Compre, M. Willemet, G. Bricteux, C. Geuzaine
//   and J-F Remacle, "Quality meshing based on STL triangulations for
//   biomedical simulations", International Journal for Numerical
//   Methods in Biomedical Engineering", 26 (7), pp. 876-889, 2010.
//
// * E. Marchandise, C. Carton de Wiart, W. G. Vos, C. Geuzaine and
//   J.-F. Remacle, "High Quality Surface Remeshing Using Harmonic
//   Maps. Part II: Surfaces with High Genus and of Large Aspect
//   Ratio", International Journal for Numerical Methods in
//   Engineering, 86 (11), pp. 1303-1321, 2011.
```

A.13 ‘t13.geo’

```
/*********************************************************************
 *
 *  Gmsh tutorial 13
 *
 *  Remeshing STL with compounds
 *
 *********************************************************************/

// Since compound geometrical compute a new parametrization, one can
// also use them to remesh STL files, even if in this case there’s
// usually only a single elementary geometrical entity per compound.

// Let’s merge the mesh that we would like to remesh. This mesh was
// reclassified ("colored") from an initial STL triangulation using
// the "Reclassify 2D" tool in Gmsh, so that we could split it along
// sharp geometrical features.
Merge "t13_data.msh";

// Since the original mesh is a bit coarse, we refine it once
```

```
RefineMesh;

// Create the topology of the discrete model
CreateTopology;

// We can now define a compound line (resp. surface) for each discrete
// line (resp. surface) in the model
ll[] = Line "*";
For j In {0 : #ll[]-1}
  Compound Line(newl) = ll[j];
EndFor
ss[] = Surface "*";
s = news;
For i In {0 : #ss[]-1}
  Compound Surface(s+i) = ss[i];
EndFor

// And we can create the volume based on the new compound entities
Surface Loop(1) = {s : s + #ss[]-1};
Volume(1) = {1};

Physical Surface(1) = {s : s + #ss[]-1};
Physical Volume(1) = 1;

// Apply a funny mesh size field, just because we can :-)
Field[1] = MathEval;
Field[1].F = "2*Sin((x+y)/5) + 3";
Background Field = 1;

Mesh.RemeshAlgorithm = 1; // (0) no split (1) automatic (2) automatic only with metis
Mesh.RemeshParametrization = 7; // (0) harmonic (1) conformal spectral (7) conformal finite
Geometry.HideCompounds = 0; // don't hide the compound entities
```

A.14 't14.geo'

```
/*********************************************************************
 *
 *  Gmsh tutorial 14
 *
 *  Homology and cohomology computation
 *
 *********************************************************************/

// Homology computation in Gmsh finds representative chains of
// (relative) (co)homology space bases using a mesh of a model.
// The representative basis chains are stored in the mesh as
// physical groups of Gmsh, one for each chain.
// Create an example geometry

m = 0.5; // mesh characteristic length
h = 2; // height in the z-direction

Point(1) = {0, 0, 0, m};   Point(2) = {10, 0, 0, m};
Point(3) = {10, 10, 0, m}; Point(4) = {0, 10, 0, m};
Point(5) = {4, 4, 0, m};   Point(6) = {6, 4, 0, m};
Point(7) = {6, 6, 0, m};   Point(8) = {4, 6, 0, m};

Point(9) = {2, 0, 0, m};   Point(10) = {8, 0, 0, m};
Point(11) = {2, 10, 0, m}; Point(12) = {8, 10, 0, m};

Line(1) = {1, 9};  Line(2) = {9, 10}; Line(3) = {10, 2};
Line(4) = {2, 3};  Line(5) = {3, 12}; Line(6) = {12, 11};
Line(7) = {11, 4}; Line(8) = {4, 1};  Line(9) = {5, 6};
Line(10) = {6, 7}; Line(11) = {7, 8}; Line(12) = {8, 5};

Line Loop(13) = {6, 7, 8, 1, 2, 3, 4, 5};
Line Loop(14) = {11, 12, 9, 10};
Plane Surface(15) = {13, 14};

Extrude {0, 0, h}{ Surface{15}; }

// Create physical groups, which are used to define the domain of the
// (co)homology computation and the subdomain of the relative (co)homology
// computation.

// Whole domain
Physical Volume(1) = {1};

// Four "terminals" of the model
Physical Surface(70) = {36};
Physical Surface(71) = {44};
Physical Surface(72) = {52};
Physical Surface(73) = {60};

// Whole domain surface
bnd[] = Boundary{ Volume{1}; };
Physical Surface(80) = bnd[];

// Complement of the domain surface respect to the four terminals
bnd[] -= {36, 44, 52, 60};
Physical Surface(75) = bnd[];

// Find bases for relative homology spaces of
// the domain modulo the four terminals.
Homology {{1}, {70, 71, 72, 73}};

// Find homology space bases isomorphic to the previous bases:
// homology spaces modulo the non-terminal domain surface,
// a.k.a the thin cuts.
Homology {{1}, {75}};

// Find cohomology space bases isomorphic to the previous bases:
// cohomology spaces of the domain modulo the four terminals,
// a.k.a the thick cuts.
Cohomology {{1}, {70, 71, 72, 73}};

// More examples:
//  Homology {1};
//  Homology;
//  Homology {{1}, {80}};
//  Homology {{}, {80}};
```

Appendix B Options

This appendix lists all the available options. Gmsh's default behavior is to save some of these options in a per-user "session resource" file (cf. "Saved in: `General.SessionFileName`" in the lists below) every time Gmsh is shut down. This permits for example to automatically remember the size and location of the windows or which fonts to use. A second set of options can be saved (automatically or manually with the 'File->Save Options->As Default' menu) in a per-user "option" file (cf. "Saved in: `General.OptionsFileName`" in the lists below), automatically loaded by Gmsh every time it starts up. Finally, other options are only saved to disk manually, either by explicitely saving an option file with 'File->Save As', or when saving per-model options with 'File->Save Options->For Current File' (cf. "Saved in: `-`" in the lists below).

To reset all options to their default values, use the 'Restore default options' button in 'Tools->Options->General->Advanced', or erase the `General.SessionFileName` and `General.OptionsFileName` files by hand.

B.1 General options list

`General.AxesFormatX`
: Number format for X-axis (in standard C form)
: Default value: `"%.3g"`
: Saved in: `General.OptionsFileName`

`General.AxesFormatY`
: Number format for Y-axis (in standard C form)
: Default value: `"%.3g"`
: Saved in: `General.OptionsFileName`

`General.AxesFormatZ`
: Number format for Z-axis (in standard C form)
: Default value: `"%.3g"`
: Saved in: `General.OptionsFileName`

`General.AxesLabelX`
: X-axis label
: Default value: `""`
: Saved in: `General.OptionsFileName`

`General.AxesLabelY`
: Y-axis label
: Default value: `""`
: Saved in: `General.OptionsFileName`

`General.AxesLabelZ`
: Z-axis label
: Default value: `""`
: Saved in: `General.OptionsFileName`

`General.BackgroundImageFileName`
: Background image file in JPEG or PNG format
  Default value: `""`
  Saved in: `General.OptionsFileName`

`General.DefaultFileName`
: Default project file name
  Default value: `"untitled.geo"`
  Saved in: `General.OptionsFileName`

`General.Display`
: X server to use (only for Unix versions)
  Default value: `""`
  Saved in: `-`

`General.ErrorFileName`
: File into which the log is saved if a fatal error occurs
  Default value: `".gmsh-errors"`
  Saved in: `General.OptionsFileName`

`General.FileName`
: Current project file name (read-only)
  Default value: `""`
  Saved in: `-`

`General.FltkTheme`
: FLTK user interface theme (try e.g. plastic or gtk+)
  Default value: `""`
  Saved in: `General.OptionsFileName`

`General.GraphicsFont`
: Font used in the graphic window
  Default value: `"Helvetica"`
  Saved in: `General.OptionsFileName`

`General.GraphicsFontTitle`
: Font used in the graphic window for titles
  Default value: `"Helvetica"`
  Saved in: `General.OptionsFileName`

`General.OptionsFileName`
: Option file created with 'Tools->Options->Save'; automatically read on startup
  Default value: `".gmsh-options"`
  Saved in: `General.SessionFileName`

`General.RecentFile0`
: Most recent opened file
  Default value: `"untitled.geo"`
  Saved in: `General.SessionFileName`

`General.RecentFile1`
: 2nd most recent opened file
Default value: `"untitled.geo"`
Saved in: `General.SessionFileName`

`General.RecentFile2`
: 3rd most recent opened file
Default value: `"untitled.geo"`
Saved in: `General.SessionFileName`

`General.RecentFile3`
: 4th most recent opened file
Default value: `"untitled.geo"`
Saved in: `General.SessionFileName`

`General.RecentFile4`
: 5th most recent opened file
Default value: `"untitled.geo"`
Saved in: `General.SessionFileName`

`General.SessionFileName`
: Option file into which session specific information is saved; automatically read on startup
Default value: `".gmshrc"`
Saved in: `-`

`General.TextEditor`
: System command to launch a text editor
Default value: `"open -t %s"`
Saved in: `General.OptionsFileName`

`General.TmpFileName`
: Temporary file used by the geometry module
Default value: `".gmsh-tmp"`
Saved in: `General.SessionFileName`

`General.WatchFilePattern`
: Pattern of files to merge as they become available
Default value: `""`
Saved in: `-`

`General.AlphaBlending`
: Enable alpha blending (transparency) in post-processing views
Default value: `1`
Saved in: `General.OptionsFileName`

`General.Antialiasing`
: Use multisample antialiasing (will slow down rendering)
Default value: `0`
Saved in: `General.OptionsFileName`

`General.ArrowHeadRadius`
Relative radius of arrow head
Default value: `0.12`
Saved in: `General.OptionsFileName`

`General.ArrowStemLength`
Relative length of arrow stem
Default value: `0.56`
Saved in: `General.OptionsFileName`

`General.ArrowStemRadius`
Relative radius of arrow stem
Default value: `0.02`
Saved in: `General.OptionsFileName`

`General.Axes`
Axes (0=none, 1=simple axes, 2=box, 3=full grid, 4=open grid, 5=ruler)
Default value: `0`
Saved in: `General.OptionsFileName`

`General.AxesMikado`
Mikado axes style
Default value: `0`
Saved in: `General.OptionsFileName`

`General.AxesAutoPosition`
Position the axes automatically
Default value: `1`
Saved in: `General.OptionsFileName`

`General.AxesForceValue`
Force values on axes (otherwise use natural coordinates)
Default value: `0`
Saved in: `General.OptionsFileName`

`General.AxesMaxX`
Maximum X-axis coordinate
Default value: `1`
Saved in: `General.OptionsFileName`

`General.AxesMaxY`
Maximum Y-axis coordinate
Default value: `1`
Saved in: `General.OptionsFileName`

`General.AxesMaxZ`
Maximum Z-axis coordinate
Default value: `1`
Saved in: `General.OptionsFileName`

`General.AxesMinX`
: Minimum X-axis coordinate
  Default value: `0`
  Saved in: `General.OptionsFileName`

`General.AxesMinY`
: Minimum Y-axis coordinate
  Default value: `0`
  Saved in: `General.OptionsFileName`

`General.AxesMinZ`
: Minimum Z-axis coordinate
  Default value: `0`
  Saved in: `General.OptionsFileName`

`General.AxesTicsX`
: Number of tics on the X-axis
  Default value: `5`
  Saved in: `General.OptionsFileName`

`General.AxesTicsY`
: Number of tics on the Y-axis
  Default value: `5`
  Saved in: `General.OptionsFileName`

`General.AxesTicsZ`
: Number of tics on the Z-axis
  Default value: `5`
  Saved in: `General.OptionsFileName`

`General.AxesValueMaxX`
: Maximum X-axis forced value
  Default value: `1`
  Saved in: `General.OptionsFileName`

`General.AxesValueMaxY`
: Maximum Y-axis forced value
  Default value: `1`
  Saved in: `General.OptionsFileName`

`General.AxesValueMaxZ`
: Maximum Z-axis forced value
  Default value: `1`
  Saved in: `General.OptionsFileName`

`General.AxesValueMinX`
: Minimum X-axis forced value
  Default value: `0`
  Saved in: `General.OptionsFileName`

`General.AxesValueMinY`
: Minimum Y-axis forced value
Default value: `0`
Saved in: `General.OptionsFileName`

`General.AxesValueMinZ`
: Minimum Z-axis forced value
Default value: `0`
Saved in: `General.OptionsFileName`

`General.BackgroundGradient`
: Draw background gradient (0=none, 1=vertical, 2=horizontal, 3=radial)
Default value: `1`
Saved in: `General.OptionsFileName`

`General.BackgroundImagePositionX`
: X position (in pixels) of background image (< 0: measure from right edge; >= 1e5: centered)
Default value: `100000`
Saved in: `General.OptionsFileName`

`General.BackgroundImagePositionY`
: Y position (in pixels) of background image (< 0: measure from bottom edge; >= 1e5: centered)
Default value: `100000`
Saved in: `General.OptionsFileName`

`General.Camera`
: Enable camera view mode
Default value: `0`
Saved in: `General.OptionsFileName`

`General.CameraAperture`
: Camera aperture in degrees
Default value: `40`
Saved in: `General.OptionsFileName`

`General.CameraEyeSeparationRatio`
: Eye separation ratio in % for stereo rendering
Default value: `1.5`
Saved in: `General.OptionsFileName`

`General.CameraFocalLengthRatio`
: Camera Focal length ratio
Default value: `1`
Saved in: `General.OptionsFileName`

`General.Clip0A`
: First coefficient in equation for clipping plane 0 (‘A’ in ‘AX+BY+CZ+D=0’)
Default value: `1`
Saved in: `-`

`General.Clip0B`
Second coefficient in equation for clipping plane 0 (‘B’ in ‘AX+BY+CZ+D=0’)
Default value: `0`
Saved in: `-`

`General.Clip0C`
Third coefficient in equation for clipping plane 0 (‘C’ in ‘AX+BY+CZ+D=0’)
Default value: `0`
Saved in: `-`

`General.Clip0D`
Fourth coefficient in equation for clipping plane 0 (‘D’ in ‘AX+BY+CZ+D=0’)
Default value: `0`
Saved in: `-`

`General.Clip1A`
First coefficient in equation for clipping plane 1
Default value: `0`
Saved in: `-`

`General.Clip1B`
Second coefficient in equation for clipping plane 1
Default value: `1`
Saved in: `-`

`General.Clip1C`
Third coefficient in equation for clipping plane 1
Default value: `0`
Saved in: `-`

`General.Clip1D`
Fourth coefficient in equation for clipping plane 1
Default value: `0`
Saved in: `-`

`General.Clip2A`
First coefficient in equation for clipping plane 2
Default value: `0`
Saved in: `-`

`General.Clip2B`
Second coefficient in equation for clipping plane 2
Default value: `0`
Saved in: `-`

`General.Clip2C`
Third coefficient in equation for clipping plane 2
Default value: `1`
Saved in: `-`

`General.Clip2D`
Fourth coefficient in equation for clipping plane 2
Default value: `0`
Saved in: `-`

`General.Clip3A`
First coefficient in equation for clipping plane 3
Default value: `-1`
Saved in: `-`

`General.Clip3B`
Second coefficient in equation for clipping plane 3
Default value: `0`
Saved in: `-`

`General.Clip3C`
Third coefficient in equation for clipping plane 3
Default value: `0`
Saved in: `-`

`General.Clip3D`
Fourth coefficient in equation for clipping plane 3
Default value: `1`
Saved in: `-`

`General.Clip4A`
First coefficient in equation for clipping plane 4
Default value: `0`
Saved in: `-`

`General.Clip4B`
Second coefficient in equation for clipping plane 4
Default value: `-1`
Saved in: `-`

`General.Clip4C`
Third coefficient in equation for clipping plane 4
Default value: `0`
Saved in: `-`

`General.Clip4D`
Fourth coefficient in equation for clipping plane 4
Default value: `1`
Saved in: `-`

`General.Clip5A`
First coefficient in equation for clipping plane 5
Default value: `0`
Saved in: `-`

`General.Clip5B`
: Second coefficient in equation for clipping plane 5
Default value: `0`
Saved in: `-`

`General.Clip5C`
: Third coefficient in equation for clipping plane 5
Default value: `-1`
Saved in: `-`

`General.Clip5D`
: Fourth coefficient in equation for clipping plane 5
Default value: `1`
Saved in: `-`

`General.ClipFactor`
: Near and far clipping plane distance factor (decrease value for better z-buffer resolution)
Default value: `5`
Saved in: `-`

`General.ClipOnlyDrawIntersectingVolume`
: Only draw layer of elements that intersect the clipping plane
Default value: `0`
Saved in: `General.OptionsFileName`

`General.ClipOnlyVolume`
: Only clip volume elements
Default value: `0`
Saved in: `General.OptionsFileName`

`General.ClipPositionX`
: Horizontal position (in pixels) of the upper left corner of the clipping planes window
Default value: `650`
Saved in: `General.SessionFileName`

`General.ClipPositionY`
: Vertical position (in pixels) of the upper left corner of the clipping planes window
Default value: `150`
Saved in: `General.SessionFileName`

`General.ClipWholeElements`
: Clip whole elements
Default value: `0`
Saved in: `General.OptionsFileName`

`General.ColorScheme`
: Default color scheme (0=dark, 1=light or 2=grayscale)
Default value: `1`
Saved in: `General.OptionsFileName`

`General.ConfirmOverwrite`
: Ask confirmation before overwriting files?
Default value: `1`
Saved in: `General.OptionsFileName`

`General.ContextPositionX`
: Horizontal position (in pixels) of the upper left corner of the contextual windows
Default value: `650`
Saved in: `General.SessionFileName`

`General.ContextPositionY`
: Vertical position (in pixels) of the upper left corner of the contextual windows
Default value: `150`
Saved in: `General.SessionFileName`

`General.DetachedMenu`
: Should the menu window be detached from the graphic window?
Default value: `0`
Saved in: `General.SessionFileName`

`General.DisplayBorderFactor`
: Border factor for model display (0: model fits window size exactly)
Default value: `0.2`
Saved in: `General.OptionsFileName`

`General.DoubleBuffer`
: Use a double buffered graphic window (on Unix, should be set to 0 when working on a remote host without GLX)
Default value: `1`
Saved in: `General.OptionsFileName`

`General.DrawBoundingBoxes`
: Draw bounding boxes
Default value: `0`
Saved in: `General.OptionsFileName`

`General.ExpertMode`
: Enable expert mode (to disable all the messages meant for inexperienced users)
Default value: `0`
Saved in: `General.OptionsFileName`

`General.ExtraPositionX`
: Horizontal position (in pixels) of the upper left corner of the generic extra window
Default value: `650`
Saved in: `General.SessionFileName`

`General.ExtraPositionY`
: Vertical position (in pixels) of the upper left corner of the generic extra window
Default value: `350`
Saved in: `General.SessionFileName`

`General.ExtraHeight`
: Height (in pixels) of the generic extra window
  Default value: `100`
  Saved in: `General.SessionFileName`

`General.ExtraWidth`
: Width (in pixels) of the generic extra window
  Default value: `100`
  Saved in: `General.SessionFileName`

`General.FastRedraw`
: Draw simplified model while rotating, panning and zooming
  Default value: `0`
  Saved in: `General.OptionsFileName`

`General.FieldPositionX`
: Horizontal position (in pixels) of the upper left corner of the field window
  Default value: `650`
  Saved in: `General.SessionFileName`

`General.FieldPositionY`
: Vertical position (in pixels) of the upper left corner of the field window
  Default value: `550`
  Saved in: `General.SessionFileName`

`General.FieldHeight`
: Height (in pixels) of the field window
  Default value: `320`
  Saved in: `General.SessionFileName`

`General.FieldWidth`
: Width (in pixels) of the field window
  Default value: `420`
  Saved in: `General.SessionFileName`

`General.FileChooserPositionX`
: Horizontal position (in pixels) of the upper left corner of the file chooser windows
  Default value: `200`
  Saved in: `General.SessionFileName`

`General.FileChooserPositionY`
: Vertical position (in pixels) of the upper left corner of the file chooser windows
  Default value: `200`
  Saved in: `General.SessionFileName`

`General.FontSize`
: Size of the font in the user interface (-1=automatic)
  Default value: `-1`
  Saved in: `General.OptionsFileName`

`General.GraphicsFontSize`
: Size of the font in the graphic window
Default value: `15`
Saved in: `General.OptionsFileName`

`General.GraphicsFontSizeTitle`
: Size of the font in the graphic window for titles
Default value: `18`
Saved in: `General.OptionsFileName`

`General.GraphicsHeight`
: Height (in pixels) of the graphic window
Default value: `600`
Saved in: `General.SessionFileName`

`General.GraphicsPositionX`
: Horizontal position (in pixels) of the upper left corner of the graphic window
Default value: `50`
Saved in: `General.SessionFileName`

`General.GraphicsPositionY`
: Vertical position (in pixels) of the upper left corner of the graphic window
Default value: `50`
Saved in: `General.SessionFileName`

`General.GraphicsWidth`
: Width (in pixels) of the graphic window
Default value: `800`
Saved in: `General.SessionFileName`

`General.InitialModule`
: Module launched on startup (0=automatic, 1=geometry, 2=mesh, 3=solver, 4=post-processing)
Default value: `0`
Saved in: `General.OptionsFileName`

`General.Light0`
: Enable light source 0
Default value: `1`
Saved in: `General.OptionsFileName`

`General.Light0X`
: X position of light source 0
Default value: `0.65`
Saved in: `General.OptionsFileName`

`General.Light0Y`
: Y position of light source 0
Default value: `0.65`
Saved in: `General.OptionsFileName`

`General.Light0Z`
: Z position of light source 0
  Default value: `1`
  Saved in: `General.OptionsFileName`

`General.Light0W`
: Divisor of the X, Y and Z coordinates of light source 0 (W=0 means infinitely far source)
  Default value: `0`
  Saved in: `General.OptionsFileName`

`General.Light1`
: Enable light source 1
  Default value: `0`
  Saved in: `General.OptionsFileName`

`General.Light1X`
: X position of light source 1
  Default value: `0.5`
  Saved in: `General.OptionsFileName`

`General.Light1Y`
: Y position of light source 1
  Default value: `0.3`
  Saved in: `General.OptionsFileName`

`General.Light1Z`
: Z position of light source 1
  Default value: `1`
  Saved in: `General.OptionsFileName`

`General.Light1W`
: Divisor of the X, Y and Z coordinates of light source 1 (W=0 means infinitely far source)
  Default value: `0`
  Saved in: `General.OptionsFileName`

`General.Light2`
: Enable light source 2
  Default value: `0`
  Saved in: `General.OptionsFileName`

`General.Light2X`
: X position of light source 2
  Default value: `0.5`
  Saved in: `General.OptionsFileName`

`General.Light2Y`
: Y position of light source 2
  Default value: `0.3`
  Saved in: `General.OptionsFileName`

`General.Light2Z`
: Z position of light source 2
Default value: `1`
Saved in: `General.OptionsFileName`

`General.Light2W`
: Divisor of the X, Y and Z coordinates of light source 2 (W=0 means infinitely far source)
Default value: `0`
Saved in: `General.OptionsFileName`

`General.Light3`
: Enable light source 3
Default value: `0`
Saved in: `General.OptionsFileName`

`General.Light3X`
: X position of light source 3
Default value: `0.5`
Saved in: `General.OptionsFileName`

`General.Light3Y`
: Y position of light source 3
Default value: `0.3`
Saved in: `General.OptionsFileName`

`General.Light3Z`
: Z position of light source 3
Default value: `1`
Saved in: `General.OptionsFileName`

`General.Light3W`
: Divisor of the X, Y and Z coordinates of light source 3 (W=0 means infinitely far source)
Default value: `0`
Saved in: `General.OptionsFileName`

`General.Light4`
: Enable light source 4
Default value: `0`
Saved in: `General.OptionsFileName`

`General.Light4X`
: X position of light source 4
Default value: `0.5`
Saved in: `General.OptionsFileName`

`General.Light4Y`
: Y position of light source 4
Default value: `0.3`
Saved in: `General.OptionsFileName`

`General.Light4Z`
Z position of light source 4
Default value: `1`
Saved in: `General.OptionsFileName`

`General.Light4W`
Divisor of the X, Y and Z coordinates of light source 4 (W=0 means infinitely far source)
Default value: `0`
Saved in: `General.OptionsFileName`

`General.Light5`
Enable light source 5
Default value: `0`
Saved in: `General.OptionsFileName`

`General.Light5X`
X position of light source 5
Default value: `0.5`
Saved in: `General.OptionsFileName`

`General.Light5Y`
Y position of light source 5
Default value: `0.3`
Saved in: `General.OptionsFileName`

`General.Light5Z`
Z position of light source 5
Default value: `1`
Saved in: `General.OptionsFileName`

`General.Light5W`
Divisor of the X, Y and Z coordinates of light source 5 (W=0 means infinitely far source)
Default value: `0`
Saved in: `General.OptionsFileName`

`General.LineWidth`
Display width of lines (in pixels)
Default value: `1`
Saved in: `General.OptionsFileName`

`General.ManipulatorPositionX`
Horizontal position (in pixels) of the upper left corner of the manipulator window
Default value: `650`
Saved in: `General.SessionFileName`

`General.ManipulatorPositionY`
Vertical position (in pixels) of the upper left corner of the manipulator window
Default value: `150`
Saved in: `General.SessionFileName`

`General.MaxX`
: Maximum model coordinate along the X-axis (read-only)
Default value: `0`
Saved in: `-`

`General.MaxY`
: Maximum model coordinate along the Y-axis (read-only)
Default value: `0`
Saved in: `-`

`General.MaxZ`
: Maximum model coordinate along the Z-axis (read-only)
Default value: `0`
Saved in: `-`

`General.MenuWidth`
: Width (in pixels) of the menu tree
Default value: `200`
Saved in: `General.SessionFileName`

`General.MenuHeight`
: Height (in pixels) of the (detached) menu tree
Default value: `200`
Saved in: `General.SessionFileName`

`General.MenuPositionX`
: Horizontal position (in pixels) of the (detached) menu tree
Default value: `400`
Saved in: `General.SessionFileName`

`General.MenuPositionY`
: Vertical position (in pixels) of the (detached) menu tree
Default value: `400`
Saved in: `General.SessionFileName`

`General.MessageHeight`
: Height (in pixels) of the message console when it is visible (should be > 0)
Default value: `300`
Saved in: `General.SessionFileName`

`General.MinX`
: Minimum model coordinate along the X-axis (read-only)
Default value: `0`
Saved in: `-`

`General.MinY`
: Minimum model coordinate along the Y-axis (read-only)
Default value: `0`
Saved in: `-`

`General.MinZ`
Minimum model coordinate along the Z-axis (read-only)
Default value: `0`
Saved in: `-`

`General.MouseHoverMeshes`
Enable mouse hover on meshes
Default value: `0`
Saved in: `General.OptionsFileName`

`General.MouseSelection`
Enable mouse selection
Default value: `1`
Saved in: `General.OptionsFileName`

`General.NonModalWindows`
Force all control windows to be on top of the graphic window ("non-modal")
Default value: `1`
Saved in: `General.SessionFileName`

`General.NoPopup`
Disable interactive dialog windows in scripts (and use default values instead)
Default value: `0`
Saved in: `General.OptionsFileName`

`General.OptionsPositionX`
Horizontal position (in pixels) of the upper left corner of the option window
Default value: `650`
Saved in: `General.SessionFileName`

`General.OptionsPositionY`
Vertical position (in pixels) of the upper left corner of the option window
Default value: `150`
Saved in: `General.SessionFileName`

`General.Orthographic`
Orthographic projection mode (0=perspective projection)
Default value: `1`
Saved in: `General.OptionsFileName`

`General.PluginPositionX`
Horizontal position (in pixels) of the upper left corner of the plugin window
Default value: `650`
Saved in: `General.SessionFileName`

`General.PluginPositionY`
Vertical position (in pixels) of the upper left corner of the plugin window
Default value: `550`
Saved in: `General.SessionFileName`

`General.PluginHeight`
: Height (in pixels) of the plugin window
  Default value: `320`
  Saved in: `General.SessionFileName`

`General.PluginWidth`
: Width (in pixels) of the plugin window
  Default value: `420`
  Saved in: `General.SessionFileName`

`General.PointSize`
: Display size of points (in pixels)
  Default value: `3`
  Saved in: `General.OptionsFileName`

`General.PolygonOffsetAlwaysOn`
: Always apply polygon offset, instead of trying to detect when it is required
  Default value: `0`
  Saved in: `General.OptionsFileName`

`General.PolygonOffsetFactor`
: Polygon offset factor (offset = factor * DZ + r * units)
  Default value: `0.5`
  Saved in: `General.OptionsFileName`

`General.PolygonOffsetUnits`
: Polygon offset units (offset = factor * DZ + r * units)
  Default value: `1`
  Saved in: `General.OptionsFileName`

`General.ProgressMeterStep`
: Increment (in percent) of the progress meter bar
  Default value: `20`
  Saved in: `General.OptionsFileName`

`General.QuadricSubdivisions`
: Number of subdivisions used to draw points or lines as spheres or cylinders
  Default value: `6`
  Saved in: `General.OptionsFileName`

`General.RotationX`
: First Euler angle (used if Trackball=0)
  Default value: `0`
  Saved in: `-`

`General.RotationY`
: Second Euler angle (used if Trackball=0)
  Default value: `0`
  Saved in: `-`

`General.RotationZ`
: Third Euler angle (used if Trackball=0)
Default value: `0`
Saved in: `-`

`General.RotationCenterGravity`
: Rotate around the (pseudo) center of mass instead of (RotationCenterX, RotationCenterY, RotationCenterZ)
Default value: `1`
Saved in: `General.OptionsFileName`

`General.RotationCenterX`
: X coordinate of the center of rotation
Default value: `0`
Saved in: `-`

`General.RotationCenterY`
: Y coordinate of the center of rotation
Default value: `0`
Saved in: `-`

`General.RotationCenterZ`
: Z coordinate of the center of rotation
Default value: `0`
Saved in: `-`

`General.SaveOptions`
: Automatically save current options in General.OptionsFileName (1) or per model (2)each time you quit Gmsh?
Default value: `0`
Saved in: `General.SessionFileName`

`General.SaveSession`
: Automatically save session specific information in General.SessionFileName each time you quit Gmsh?
Default value: `1`
Saved in: `General.SessionFileName`

`General.ScaleX`
: X-axis scale factor
Default value: `1`
Saved in: `-`

`General.ScaleY`
: Y-axis scale factor
Default value: `1`
Saved in: `-`

`General.ScaleZ`
: Z-axis scale factor
Default value: `1`
Saved in: `-`

`General.Shininess`
: Material shininess
Default value: `0.4`
Saved in: `General.OptionsFileName`

`General.ShininessExponent`
: Material shininess exponent (between 0 and 128)
Default value: `40`
Saved in: `General.OptionsFileName`

`General.SmallAxes`
: Display the small axes
Default value: `1`
Saved in: `General.OptionsFileName`

`General.SmallAxesPositionX`
: X position (in pixels) of small axes (< 0: measure from right edge; >= 1e5: centered)
Default value: `-60`
Saved in: `General.OptionsFileName`

`General.SmallAxesPositionY`
: Y position (in pixels) of small axes (< 0: measure from bottom edge; >= 1e5: centered)
Default value: `-40`
Saved in: `General.OptionsFileName`

`General.SmallAxesSize`
: Size (in pixels) of small axes
Default value: `30`
Saved in: `General.OptionsFileName`

`General.StatisticsPositionX`
: Horizontal position (in pixels) of the upper left corner of the statistic window
Default value: `650`
Saved in: `General.SessionFileName`

`General.StatisticsPositionY`
: Vertical position (in pixels) of the upper left corner of the statistic window
Default value: `150`
Saved in: `General.SessionFileName`

`General.Stereo`
: Use stereo rendering
Default value: `0`
Saved in: `General.OptionsFileName`

`General.SystemMenuBar`
: Use the system menu bar on Mac OS X?
Default value: `1`
Saved in: `General.SessionFileName`

`General.Terminal`
: Should information be printed on the terminal (if available)?
Default value: `0`
Saved in: `General.OptionsFileName`

`General.Tooltips`
: Show tooltips in the user interface
Default value: `1`
Saved in: `General.OptionsFileName`

`General.Trackball`
: Use trackball rotation mode
Default value: `1`
Saved in: `General.OptionsFileName`

`General.TrackballHyperbolicSheet`
: Use hyperbolic sheet away from trackball center for z-rotations
Default value: `1`
Saved in: `General.OptionsFileName`

`General.TrackballQuaternion0`
: First trackball quaternion component (used if General.Trackball=1)
Default value: `0`
Saved in: `-`

`General.TrackballQuaternion1`
: Second trackball quaternion component (used if General.Trackball=1)
Default value: `0`
Saved in: `-`

`General.TrackballQuaternion2`
: Third trackball quaternion component (used if General.Trackball=1)
Default value: `0`
Saved in: `-`

`General.TrackballQuaternion3`
: Fourth trackball quaternion component (used if General.Trackball=1)
Default value: `1`
Saved in: `-`

`General.TranslationX`
: X-axis translation (in model units)
Default value: `0`
Saved in: `-`

`General.TranslationY`
: Y-axis translation (in model units)
Default value: `0`
Saved in: `-`

`General.TranslationZ`
: Z-axis translation (in model units)
Default value: `0`
Saved in: `-`

`General.VectorType`
: Default vector display type (for normals, etc.)
Default value: `4`
Saved in: `General.OptionsFileName`

`General.Verbosity`
: Level of information printed during processing (0=no information)
Default value: `5`
Saved in: `General.OptionsFileName`

`General.VisibilityPositionX`
: Horizontal position (in pixels) of the upper left corner of the visibility window
Default value: `650`
Saved in: `General.SessionFileName`

`General.VisibilityPositionY`
: Vertical position (in pixels) of the upper left corner of the visibility window
Default value: `150`
Saved in: `General.SessionFileName`

`General.ZoomFactor`
: Middle mouse button zoom acceleration factor
Default value: `4`
Saved in: `General.OptionsFileName`

`General.Color.Background`
: Background color
Default value: `{255,255,255}`
Saved in: `General.OptionsFileName`

`General.Color.BackgroundGradient`
: Background gradient color
Default value: `{128,147,255}`
Saved in: `General.OptionsFileName`

`General.Color.Foreground`
: Foreground color
Default value: `{85,85,85}`
Saved in: `General.OptionsFileName`

`General.Color.Text`
: Text color
Default value: `{0,0,0}`
Saved in: `General.OptionsFileName`

`General.Color.Axes`
: Axes color
: Default value: `{0,0,0}`
: Saved in: `General.OptionsFileName`

`General.Color.SmallAxes`
: Small axes color
: Default value: `{0,0,0}`
: Saved in: `General.OptionsFileName`

`General.Color.AmbientLight`
: Ambient light color
: Default value: `{25,25,25}`
: Saved in: `General.OptionsFileName`

`General.Color.DiffuseLight`
: Diffuse light color
: Default value: `{255,255,255}`
: Saved in: `General.OptionsFileName`

`General.Color.SpecularLight`
: Specular light color
: Default value: `{255,255,255}`
: Saved in: `General.OptionsFileName`

`Print.Background`
: Print background?
: Default value: `0`
: Saved in: `General.OptionsFileName`

`Print.CompositeWindows`
: Composite all window tiles in the same output image (for bitmap output only)
: Default value: `0`
: Saved in: `General.OptionsFileName`

`Print.DeleteTemporaryFiles`
: Delete temporary files used during printing
: Default value: `1`
: Saved in: `General.OptionsFileName`

`Print.EpsBestRoot`
: Try to minimize primitive splitting in BSP tree sorted PostScript/PDF output
: Default value: `1`
: Saved in: `General.OptionsFileName`

`Print.EpsCompress`
: Compress PostScript/PDF output using zlib
: Default value: `0`
: Saved in: `General.OptionsFileName`

`Print.EpsLineWidthFactor`
: Width factor for lines in PostScript/PDF output
Default value: `1`
Saved in: `General.OptionsFileName`

`Print.EpsOcclusionCulling`
: Cull occluded primitives (to reduce PostScript/PDF file size)
Default value: `1`
Saved in: `General.OptionsFileName`

`Print.EpsPointSizeFactor`
: Size factor for points in PostScript/PDF output
Default value: `1`
Saved in: `General.OptionsFileName`

`Print.EpsPS3Shading`
: Enable PostScript Level 3 shading
Default value: `0`
Saved in: `General.OptionsFileName`

`Print.EpsQuality`
: PostScript/PDF quality (0=bitmap, 1=vector (simple sort), 2=vector (accurate sort), 3=vector (unsorted)
Default value: `1`
Saved in: `General.OptionsFileName`

`Print.Format`
: File format (10=automatic)
Default value: `10`
Saved in: `General.OptionsFileName`

`Print.GeoLabels`
: Save labels in unrolled Gmsh geometries
Default value: `1`
Saved in: `General.OptionsFileName`

`Print.GeoOnlyPhysicals`
: Only save entities that belong to physical groups
Default value: `0`
Saved in: `General.OptionsFileName`

`Print.GifDither`
: Apply dithering to GIF output
Default value: `0`
Saved in: `General.OptionsFileName`

`Print.GifInterlace`
: Interlace GIF output
Default value: `0`
Saved in: `General.OptionsFileName`

`Print.GifSort`
Sort the colormap in GIF output
Default value: `1`
Saved in: `General.OptionsFileName`

`Print.GifTransparent`
Output transparent GIF image
Default value: `0`
Saved in: `General.OptionsFileName`

`Print.Height`
Height of printed image; use (possibly scaled) current height if < 0
Default value: `-1`
Saved in: `General.OptionsFileName`

`Print.JpegQuality`
JPEG quality (between 1 and 100)
Default value: `100`
Saved in: `General.OptionsFileName`

`Print.JpegSmoothing`
JPEG smoothing (between 0 and 100)
Default value: `0`
Saved in: `General.OptionsFileName`

`Print.PostElementary`
Save elementary region tags in mesh statistics exported as post-processing views
Default value: `1`
Saved in: `General.OptionsFileName`

`Print.PostElement`
Save element numbers in mesh statistics exported as post-processing views
Default value: `0`
Saved in: `General.OptionsFileName`

`Print.PostGamma`
Save Gamma quality measure in mesh statistics exported as post-processing views
Default value: `0`
Saved in: `General.OptionsFileName`

`Print.PostEta`
Save Eta quality measure in mesh statistics exported as post-processing views
Default value: `0`
Saved in: `General.OptionsFileName`

`Print.PostRho`
Save Rho quality measure in mesh statistics exported as post-processing views
Default value: `0`
Saved in: `General.OptionsFileName`

`Print.PostDisto`
: Save Disto quality measure in mesh statistics exported as post-processing views
Default value: `0`
Saved in: `General.OptionsFileName`

`Print.TexAsEquation`
: Print all TeX strings as equations
Default value: `0`
Saved in: `General.OptionsFileName`

`Print.Text`
: Print text strings?
Default value: `1`
Saved in: `General.OptionsFileName`

`Print.Width`
: Width of printed image; use (possibly scaled) current width if < 0)
Default value: `-1`
Saved in: `General.OptionsFileName`

B.2 Geometry options list

`Geometry.AutoCoherence`
: Should all duplicate entities be automatically removed?
Default value: `1`
Saved in: `General.OptionsFileName`

`Geometry.Clip`
: Enable clipping planes? (Plane[i]=2^i, i=0,...,5)
Default value: `0`
Saved in: `-`

`Geometry.CopyMeshingMethod`
: Copy meshing method (unstructured or transfinite) when duplicating geometrical entities?
Default value: `0`
Saved in: `General.OptionsFileName`

`Geometry.ExactExtrusion`
: Use exact extrusion formula in interpolations (set to 0 to allow geometrical transformations of extruded entities)
Default value: `1`
Saved in: `General.OptionsFileName`

`Geometry.ExtrudeReturnLateralEntities`
: Add lateral entities in lists returned by extrusion commands?
Default value: `1`
Saved in: `General.OptionsFileName`

`Geometry.ExtrudeSplinePoints`
: Number of control points for splines created during extrusion
Default value: `5`
Saved in: `General.OptionsFileName`

`Geometry.HideCompounds`
Hide entities that make up compound entities?
Default value: `1`
Saved in: `General.OptionsFileName`

`Geometry.HighlightOrphans`
Highlight orphan entities (lines connected to a single surface, etc.)?
Default value: `0`
Saved in: `General.OptionsFileName`

`Geometry.LabelType`
Type of entity label (1=elementary number, 2=physical number)
Default value: `1`
Saved in: `General.OptionsFileName`

`Geometry.Light`
Enable lighting for the geometry
Default value: `1`
Saved in: `General.OptionsFileName`

`Geometry.LightTwoSide`
Light both sides of surfaces (leads to slower rendering)
Default value: `1`
Saved in: `General.OptionsFileName`

`Geometry.Lines`
Display geometry curves?
Default value: `1`
Saved in: `General.OptionsFileName`

`Geometry.LineNumbers`
Display curve numbers?
Default value: `0`
Saved in: `General.OptionsFileName`

`Geometry.LineSelectWidth`
Display width of selected lines (in pixels)
Default value: `2`
Saved in: `General.OptionsFileName`

`Geometry.LineType`
Display lines as solid color segments (0), 3D cylinders (1) or tapered cylinders (2)
Default value: `0`
Saved in: `General.OptionsFileName`

`Geometry.LineWidth`
Display width of lines (in pixels)
Default value: `2`
Saved in: `General.OptionsFileName`

`Geometry.MatchGeomAndMesh`
Matches geometries and meshes.
Default value: `0`
Saved in: `General.OptionsFileName`

`Geometry.Normals`
Display size of normal vectors (in pixels)
Default value: `0`
Saved in: `General.OptionsFileName`

`Geometry.NumSubEdges`
Number of edge subdivisions between control points when displaying curves
Default value: `20`
Saved in: `General.OptionsFileName`

`Geometry.OCCFixDegenerated`
Fix degenerated edges/faces in STEP, IGES and BRep models
Default value: `0`
Saved in: `General.OptionsFileName`

`Geometry.OCCFixSmallEdges`
Fix small edges in STEP, IGES and BRep models
Default value: `0`
Saved in: `General.OptionsFileName`

`Geometry.OCCFixSmallFaces`
Fix small faces in STEP, IGES and BRep models
Default value: `0`
Saved in: `General.OptionsFileName`

`Geometry.OCCSewFaces`
Sew faces in STEP, IGES and BRep models
Default value: `0`
Saved in: `General.OptionsFileName`

`Geometry.OCCConnectFaces`
Cut and connect faces in STEP, IGES and BRep models
Default value: `0`
Saved in: `General.OptionsFileName`

`Geometry.OffsetX`
Model display offset along X-axis (in model coordinates)
Default value: `0`
Saved in: `-`

`Geometry.OffsetY`
Model display offset along Y-axis (in model coordinates)
Default value: `0`
Saved in: `-`

`Geometry.OffsetZ`
: Model display offset along Z-axis (in model coordinates)
Default value: `0`
Saved in: `-`

`Geometry.OldCircle`
: Use old circle description (compatibility option for old Gmsh geometries)
Default value: `0`
Saved in: `General.OptionsFileName`

`Geometry.OldRuledSurface`
: Use old 3-sided ruled surface interpolation (compatibility option for old Gmsh geometries)
Default value: `0`
Saved in: `General.OptionsFileName`

`Geometry.OldNewReg`
: Use old newreg definition for geometrical transformations (compatibility option for old Gmsh geometries)
Default value: `1`
Saved in: `General.OptionsFileName`

`Geometry.Points`
: Display geometry points?
Default value: `1`
Saved in: `General.OptionsFileName`

`Geometry.PointNumbers`
: Display points numbers?
Default value: `0`
Saved in: `General.OptionsFileName`

`Geometry.PointSelectSize`
: Display size of selected points (in pixels)
Default value: `5`
Saved in: `General.OptionsFileName`

`Geometry.PointSize`
: Display size of points (in pixels)
Default value: `4`
Saved in: `General.OptionsFileName`

`Geometry.PointType`
: Display points as solid color dots (0) or 3D spheres (1)
Default value: `0`
Saved in: `General.OptionsFileName`

`Geometry.ScalingFactor`
: Global geometry scaling factor
Default value: `1`
Saved in: `General.OptionsFileName`

`Geometry.OrientedPhysicals`
: Use sign of elementary entity in physical definition as orientation indicator
  Default value: `1`
  Saved in: `General.OptionsFileName`

`Geometry.SnapX`
: Snapping grid spacing along the X-axis
  Default value: `0.1`
  Saved in: `General.OptionsFileName`

`Geometry.SnapY`
: Snapping grid spacing along the Y-axis
  Default value: `0.1`
  Saved in: `General.OptionsFileName`

`Geometry.SnapZ`
: Snapping grid spacing along the Z-axis
  Default value: `0.1`
  Saved in: `General.OptionsFileName`

`Geometry.Surfaces`
: Display geometry surfaces?
  Default value: `0`
  Saved in: `General.OptionsFileName`

`Geometry.SurfaceNumbers`
: Display surface numbers?
  Default value: `0`
  Saved in: `General.OptionsFileName`

`Geometry.SurfaceType`
: Surface display type (0=cross, 1=wireframe, 2=solid)
  Default value: `2`
  Saved in: `General.OptionsFileName`

`Geometry.Tangents`
: Display size of tangent vectors (in pixels)
  Default value: `0`
  Saved in: `General.OptionsFileName`

`Geometry.Tolerance`
: Geometrical tolerance
  Default value: `1e-06`
  Saved in: `General.OptionsFileName`

`Geometry.Transform`
: Transform model display coordinates (0=no, 1=scale)
  Default value: `0`
  Saved in: `-`

`Geometry.TransformXX`
Element (1,1) of the 3x3 model display transformation matrix
Default value: `1`
Saved in: `-`

`Geometry.TransformXY`
Element (1,2) of the 3x3 model display transformation matrix
Default value: `0`
Saved in: `-`

`Geometry.TransformXZ`
Element (1,3) of the 3x3 model display transformation matrix
Default value: `0`
Saved in: `-`

`Geometry.TransformYX`
Element (2,1) of the 3x3 model display transformation matrix
Default value: `0`
Saved in: `-`

`Geometry.TransformYY`
Element (2,2) of the 3x3 model display transformation matrix
Default value: `1`
Saved in: `-`

`Geometry.TransformYZ`
Element (2,3) of the 3x3 model display transformation matrix
Default value: `0`
Saved in: `-`

`Geometry.TransformZX`
Element (3,1) of the 3x3 model display transformation matrix
Default value: `0`
Saved in: `-`

`Geometry.TransformZY`
Element (3,2) of the 3x3 model display transformation matrix
Default value: `0`
Saved in: `-`

`Geometry.TransformZZ`
Element (3,3) of the 3x3 model display transformation matrix
Default value: `1`
Saved in: `-`

`Geometry.Volumes`
Display geometry volumes? (not implemented yet)
Default value: `0`
Saved in: `General.OptionsFileName`

`Geometry.VolumeNumbers`
: Display volume numbers? (not implemented yet)
Default value: `0`
Saved in: `General.OptionsFileName`

`Geometry.Color.Points`
: Normal geometry point color
Default value: `{90,90,90}`
Saved in: `General.OptionsFileName`

`Geometry.Color.Lines`
: Normal geometry curve color
Default value: `{0,0,255}`
Saved in: `General.OptionsFileName`

`Geometry.Color.Surfaces`
: Normal geometry surface color
Default value: `{128,128,128}`
Saved in: `General.OptionsFileName`

`Geometry.Color.Volumes`
: Normal geometry volume color
Default value: `{255,255,0}`
Saved in: `General.OptionsFileName`

`Geometry.Color.Selection`
: Selected geometry color
Default value: `{255,0,0}`
Saved in: `General.OptionsFileName`

`Geometry.Color.HighlightZero`
: Highlight 0 color
Default value: `{255,0,0}`
Saved in: `General.OptionsFileName`

`Geometry.Color.HighlightOne`
: Highlight 1 color
Default value: `{255,150,0}`
Saved in: `General.OptionsFileName`

`Geometry.Color.HighlightTwo`
: Highlight 2 color
Default value: `{255,255,0}`
Saved in: `General.OptionsFileName`

`Geometry.Color.Tangents`
: Tangent geometry vectors color
Default value: `{255,255,0}`
Saved in: `General.OptionsFileName`

`Geometry.Color.Normals`
: Normal geometry vectors color
Default value: `{255,0,0}`
Saved in: `General.OptionsFileName`

`Geometry.Color.Projection`
: Projection surface color
Default value: `{0,255,0}`
Saved in: `General.OptionsFileName`

B.3 Mesh options list

`Mesh.Algorithm`
: 2D mesh algorithm (1=MeshAdapt, 2=Automatic, 5=Delaunay, 6=Frontal, 7=bamg, 8=delquad)
Default value: `2`
Saved in: `General.OptionsFileName`

`Mesh.Algorithm3D`
: 3D mesh algorithm (1=Delaunay, 4=Frontal, 5=Frontal Delaunay, 6=Frontal Hex, 7=MMG3D, 9=R-tree)
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.AngleSmoothNormals`
: Threshold angle below which normals are not smoothed
Default value: `30`
Saved in: `General.OptionsFileName`

`Mesh.AnisoMax`
: Maximum anisotropy of the mesh
Default value: `1e+33`
Saved in: `General.OptionsFileName`

`Mesh.AllowSwapAngle`
: Threshold angle (in degrees) between faces normals under which we allow an edge swap
Default value: `10`
Saved in: `General.OptionsFileName`

`Mesh.BdfFieldFormat`
: Field format for Nastran BDF files (0=free, 1=small, 2=large)
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.Binary`
: Write mesh files in binary format (if possible)
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.Bunin`
: Apply Bunin optimization on quad meshes (the parameter is the maximal size of a cavity that may be remeshed)

Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.Lloyd`
Apply lloyd optimization on surface meshes
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.SmoothCrossField`
Apply n barycentric smoothing passes to the cross field
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.CgnsImportOrder`
Enable the creation of high-order mesh from CGNS structured meshes.(1, 2, 4, 8, ...)
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.ChacoArchitecture`
(Adv. Chaco): Parallel architecture topology (0=hypercube, 1-3=mesh dimensions)
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.ChacoEigensolver`
(Adv. Chaco): Type of eigensolver for a spectral algorithm (0=Lanczos, 1=Multilevel RQI/Symmlq)
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.ChacoEigTol`
(Adv. Chaco): Tolerance of the eigensolver for spectral or multilevel-KL algorithms
Default value: `0.001`
Saved in: `General.OptionsFileName`

`Mesh.ChacoGlobalMethod`
Chaco partitioning algorithm (1=Multilevel-KL, 2=Spectral, 4=Linear, 5=Random, 6=Scattered)
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.ChacoHypercubeDim`
(Adv. Chaco): Dimensional partitioning for a hypercube topology
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.ChacoLocalMethod`
(Adv. Chaco): Local partitioning algorithm
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.ChacoMeshDim1`
: (Adv. Chaco): Number of partitions in the first dimension of a mesh topology
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.ChacoMeshDim2`
: (Adv. Chaco): Number of partitions in the second dimension of a mesh topology
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.ChacoMeshDim3`
: (Adv. Chaco): Number of partitions in the third dimension of a mesh topology
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.ChacoPartitionSection`
: (Adv. Chaco): Partition by (1=bisection, 2=quadrisection, 3=octasection
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.ChacoSeed`
: (Adv. Chaco): Seed for random number generator
Default value: `7.65432e+06`
Saved in: `General.OptionsFileName`

`Mesh.ChacoVMax`
: (Adv. Chaco): Maximum vertices in a coarse graph (for multilevel-KL algorithm and Multilevel RQI/Symmlq eigensolver)
Default value: `250`
Saved in: `General.OptionsFileName`

`Mesh.ChacoParamINTERNAL_VERTICES`
: (Adv. Chaco): Parameter INTERNAL_VERTICES
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.ChacoParamREFINE_MAP`
: (Adv. Chaco): Parameter REFINE_MAP
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.ChacoParamREFINE_PARTITION`
: (Adv. Chaco): Parameter REFINE_PARTITION
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.ChacoParamTERMINAL_PROPOGATION`
: (Adv. Chaco): Parameter TERMINAL_PROPOGATION
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.CharacteristicLengthExtendFromBoundary`
: Extend computation of mesh element sizes from the boundaries into the surfaces/volumes
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.CharacteristicLengthFactor`
: Factor applied to all mesh element sizes
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.CharacteristicLengthMin`
: Minimum mesh element size
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.CharacteristicLengthMax`
: Maximum mesh element size
Default value: `1e+22`
Saved in: `General.OptionsFileName`

`Mesh.CharacteristicLengthFromCurvature`
: Automatically compute mesh element sizes from curvature (experimental)
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.CharacteristicLengthFromPoints`
: Compute mesh element sizes from values given at geometry points
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.Clip`
: Enable clipping planes? (Plane[i]=2^i, i=0,...,5)
Default value: `0`
Saved in: `-`

`Mesh.ColorCarousel`
: Mesh coloring (0=by element type, 1=by elementary entity, 2=by physical entity, 3=by partition)
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.CpuTime`
: CPU time (in seconds) for the generation of the current mesh (read-only)
Default value: `0`
Saved in: `-`

`Mesh.DrawSkinOnly`
: Draw only the skin of 3D meshes?
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.Dual`
Display the dual mesh obtained by barycentric subdivision
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.ElementOrder`
Element order (1=linear elements, N (<6) = elements of higher order)
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.Explode`
Element shrinking factor (between 0 and 1)
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.Format`
Mesh output format (1=msh, 2=unv, 10=automatic, 19=vrml, 27=stl, 30=mesh, 31=bdf, 32=cgns, 33=med, 40=ply2)
Default value: `10`
Saved in: `General.OptionsFileName`

`Mesh.Hexahedra`
Display mesh hexahedra?
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.HighOrderNumLayers`
Number of high order mesh elements to consider for optimization.
Default value: `3`
Saved in: `-`

`Mesh.HighOrderOptimize`
Number of smoothing steps of internal edges for high order meshes
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.HighOrderPoissonRatio`
Poisson ratio of the material used in the elastic smoother for high order meshes.Must be between -1.0 and 0.5, excluded.
Default value: `0.33`
Saved in: `-`

`Mesh.HighOrderSmoothingThreshold`
Distorition threshold when choosing which high order element to optimize.
Default value: `0.5`
Saved in: `General.OptionsFileName`

`Mesh.LabelSampling`
Label sampling rate (display one label every 'LabelSampling' elements)
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.LabelType`
: Type of element label (0=element number, 1=elementary entity number, 2=physical entity number, 3=partition number, 4=coordinates)
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.LcIntegrationPrecision`
: Accuracy of evaluation of the LC field for 1D mesh generation
Default value: `1e-09`
Saved in: `General.OptionsFileName`

`Mesh.Light`
: Enable lighting for the mesh
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.LightLines`
: Enable lighting for mesh lines (element edges)
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.LightTwoSide`
: Light both sides of surfaces (leads to slower rendering)
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.Lines`
: Display mesh lines (1D elements)?
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.LineNumbers`
: Display mesh line numbers?
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.LineWidth`
: Display width of mesh lines (in pixels)
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.MeshOnlyVisible`
: Mesh only visible entities (experimental: use with caution!)
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.MetisAlgorithm`
: METIS partitioning algorithm (1=Recursive, 2=K-way, 3=Nodal weight)
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.MetisEdgeMatching`
(Adv. METIS): Determines the matching type (1=Random, 2=Heavy-Edge, 3=Sorted Heavy-Edge)
Default value: `3`
Saved in: `General.OptionsFileName`

`Mesh.MetisRefinementAlgorithm`
(Adv. METIS): Algorithm for k-way refinement (1=Random, 2=Greedy, 3=Random with minimized connectivity)
Default value: `3`
Saved in: `General.OptionsFileName`

`Mesh.MinimumCirclePoints`
Minimum number of points used to mesh a circle
Default value: `7`
Saved in: `General.OptionsFileName`

`Mesh.MinimumCurvePoints`
Minimum number of points used to mesh a (non-straight) curve
Default value: `3`
Saved in: `General.OptionsFileName`

`Mesh.MshFileVersion`
Version of the MSH file format to use
Default value: `2.2`
Saved in: `General.OptionsFileName`

`Mesh.MshFilePartitioned`
Split MSH file by mesh partition (0: no, 1: yes, 2: create physicals by partition)
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.MultiplePassesMeshes`
Do a first simple mesh and use it for complex background meshes (curvatures...)
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.PartitionHexWeight`
Weight of hexahedral element for METIS load balancing
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.PartitionPrismWeight`
Weight of prismatic element (wedge) for METIS load balancing
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.PartitionPyramidWeight`
Weight of pyramidal element for METIS load balancing
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.PartitionQuadWeight`
: Weight of quadrangle for METIS load balancing
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.PartitionTetWeight`
: Weight of tetrahedral element for METIS load balancing
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.PartitionTriWeight`
: Weight of triangle for METIS load balancing
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.NbHexahedra`
: Number of hexahedra in the current mesh (read-only)
Default value: `0`
Saved in: `-`

`Mesh.NbNodes`
: Number of nodes in the current mesh (read-only)
Default value: `0`
Saved in: `-`

`Mesh.NbPartitions`
: Number of partitions
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.NbPrisms`
: Number of prisms in the current mesh (read-only)
Default value: `0`
Saved in: `-`

`Mesh.NbPyramids`
: Number of pyramids in the current mesh (read-only)
Default value: `0`
Saved in: `-`

`Mesh.NbQuadrangles`
: Number of quadrangles in the current mesh (read-only)
Default value: `0`
Saved in: `-`

`Mesh.NbTetrahedra`
: Number of tetrahedra in the current mesh (read-only)
Default value: `0`
Saved in: `-`

`Mesh.NbTriangles`
: Number of triangles in the current mesh (read-only)
  Default value: `0`
  Saved in: `-`

`Mesh.Normals`
: Display size of normal vectors (in pixels)
  Default value: `0`
  Saved in: `General.OptionsFileName`

`Mesh.NumSubEdges`
: Number of edge subdivisions when displaying high order elements
  Default value: `2`
  Saved in: `General.OptionsFileName`

`Mesh.Optimize`
: Optimize the mesh to improve the quality of tetrahedral elements
  Default value: `0`
  Saved in: `General.OptionsFileName`

`Mesh.OptimizeNetgen`
: Optimize the mesh using Netgen to improve the quality of tetrahedral elements
  Default value: `0`
  Saved in: `General.OptionsFileName`

`Mesh.Partitioner`
: Partitioner software (1=Chacho, 2=METIS)
  Default value: `2`
  Saved in: `General.OptionsFileName`

`Mesh.Points`
: Display mesh vertices (nodes)?
  Default value: `0`
  Saved in: `General.OptionsFileName`

`Mesh.PointNumbers`
: Display mesh node numbers?
  Default value: `0`
  Saved in: `General.OptionsFileName`

`Mesh.PointSize`
: Display size of mesh vertices (in pixels)
  Default value: `4`
  Saved in: `General.OptionsFileName`

`Mesh.PointType`
: Display mesh vertices as solid color dots (0) or 3D spheres (1)
  Default value: `0`
  Saved in: `General.OptionsFileName`

`Mesh.Prisms`
Display mesh prisms?
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.Pyramids`
Display mesh pyramids?
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.Quadrangles`
Display mesh quadrangles?
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.QualityInf`
Only display elements whose quality measure is greater than QualityInf
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.QualitySup`
Only display elements whose quality measure is smaller than QualitySup
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.QualityType`
Type of quality measure (0=gamma~vol/sum_face/max_edge, 1=eta~vol^(2/3)/sum_edge^2, 2=rho~min_edge/max_edge)
Default value: `2`
Saved in: `General.OptionsFileName`

`Mesh.RadiusInf`
Only display elements whose longest edge is greater than RadiusInf
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.RadiusSup`
Only display elements whose longest edge is smaller than RadiusSup
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.RandomFactor`
Random factor used in the 2D meshing algorithm (should be increased if RandomFactor * size(triangle)/size(model) approaches machine accuracy)
Default value: `1e-09`
Saved in: `General.OptionsFileName`

`Mesh.IgnorePartitionBoundary`
Ignore partitions boundaries (0=no, 1=yes)
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.RecombinationAlgorithm`
: Mesh recombination algorithm (0=standard, 1=blossom)
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.RecombineAll`
: Apply recombination algorithm to all surfaces, ignoring per-surface spec
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.Recombine3DAll`
: Apply recombination3D algorithm to all volumes, ignoring per-volume spec
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.RemeshAlgorithm`
: Remeshing algorithm (0=no split, 1=automatic, 2=automatic only with metis)
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.RemeshParametrization`
: Remeshing using discrete parametrization (0=harmonic_circle, 1=conformal_spectral, 2=rbf, 3=harmonic_plane, 4=convex_circle, 5=convex_plane, 6=harmonic square, 7=conformal_fe
Default value: `4`
Saved in: `General.OptionsFileName`

`Mesh.RefineSteps`
: Number of refinement steps in the MeshAdapt-based 2D algorithms
Default value: `10`
Saved in: `General.OptionsFileName`

`Mesh.Remove4Triangles`
: Try to remove nodes surrounded by 4 triangles in 2D triangular meshes
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.ReverseAllNormals`
: Reverse all the mesh normals (for display)
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.SaveAll`
: Ignore Physical definitions and save all elements
Default value: `0`
Saved in: `-`

`Mesh.SaveElementTagType`
: Type of the element tag saved in mesh formats that don't support saving physical or partition ids (1=elementary, 2=physical, 3=partition)
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.SaveParametric`
: Save parametric coordinates of nodes
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.SaveGroupsOfNodes`
: Save groups of nodes for each physical line and surface (UNV mesh format only)
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.ScalingFactor`
: Global scaling factor applied to the saved mesh
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.SecondOrderExperimental`
: Use experimental code to generate second order mesh
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.SecondOrderIncomplete`
: Create incomplete second order elements? (8-node quads, 20-node hexas, etc.)
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.SecondOrderLinear`
: Should second order vertices simply be created by linear interpolation?
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.Smoothing`
: Number of smoothing steps applied to the final mesh
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.SmoothNormals`
: Smooth the mesh normals?
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.SmoothRatio`
: Ratio between mesh sizes at vertices of a same edeg (used in BAMG)
Default value: `1.8`
Saved in: `General.OptionsFileName`

`Mesh.SubdivisionAlgorithm`
: Mesh subdivision algorithm (0=none, 1=all quadrangles, 2=all hexahedra)
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.SurfaceEdges`
: Display edges of surface mesh?
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.SurfaceFaces`
: Display faces of surface mesh?
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.SurfaceNumbers`
: Display surface mesh element numbers?
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.SwitchElementTags`
: Invert elementary and physical tags when reading the mesh
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.Tangents`
: Display size of tangent vectors (in pixels)
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.Tetrahedra`
: Display mesh tetrahedra?
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.ToleranceEdgeLength`
: Skip a model edge in mesh generation if its length is less than user's defined tolerance
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.Triangles`
: Display mesh triangles?
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.VolumeEdges`
: Display edges of volume mesh?
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.VolumeFaces`
: Display faces of volume mesh?
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.VolumeNumbers`
Display volume mesh element numbers?
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.Voronoi`
Display the voronoi diagram
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.ZoneDefinition`
Method for defining a zone (0=single zone, 1=by partition, 2=by physical)
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.Color.Points`
Mesh node color
Default value: `{0,0,255}`
Saved in: `General.OptionsFileName`

`Mesh.Color.PointsSup`
Second order mesh node color
Default value: `{255,0,255}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Lines`
Mesh line color
Default value: `{0,0,0}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Triangles`
Mesh triangle color (if Mesh.ColorCarousel=0)
Default value: `{160,150,255}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Quadrangles`
Mesh quadrangle color (if Mesh.ColorCarousel=0)
Default value: `{130,120,225}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Tetrahedra`
Mesh tetrahedron color (if Mesh.ColorCarousel=0)
Default value: `{160,150,255}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Hexahedra`
Mesh hexahedron color (if Mesh.ColorCarousel=0)
Default value: `{130,120,225}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Prisms`
Mesh prism color (if Mesh.ColorCarousel=0)
Default value: `{232,210,23}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Pyramids`
Mesh pyramid color (if Mesh.ColorCarousel=0)
Default value: `{217,113,38}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Tangents`
Tangent mesh vector color
Default value: `{255,255,0}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Normals`
Normal mesh vector color
Default value: `{255,0,0}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Zero`
Color 0 in color carousel
Default value: `{255,120,0}`
Saved in: `General.OptionsFileName`

`Mesh.Color.One`
Color 1 in color carousel
Default value: `{0,255,132}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Two`
Color 2 in color carousel
Default value: `{255,160,0}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Three`
Color 3 in color carousel
Default value: `{0,255,192}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Four`
Color 4 in color carousel
Default value: `{255,200,0}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Five`
Color 5 in color carousel
Default value: `{0,216,255}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Six`
: Color 6 in color carousel
Default value: `{255,240,0}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Seven`
: Color 7 in color carousel
Default value: `{0,176,255}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Eight`
: Color 8 in color carousel
Default value: `{228,255,0}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Nine`
: Color 9 in color carousel
Default value: `{0,116,255}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Ten`
: Color 10 in color carousel
Default value: `{188,255,0}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Eleven`
: Color 11 in color carousel
Default value: `{0,76,255}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Twelve`
: Color 12 in color carousel
Default value: `{148,255,0}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Thirteen`
: Color 13 in color carousel
Default value: `{24,0,255}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Fourteen`
: Color 14 in color carousel
Default value: `{108,255,0}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Fifteen`
: Color 15 in color carousel
Default value: `{84,0,255}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Sixteen`
: Color 16 in color carousel
  Default value: `{68,255,0}`
  Saved in: `General.OptionsFileName`

`Mesh.Color.Seventeen`
: Color 17 in color carousel
  Default value: `{104,0,255}`
  Saved in: `General.OptionsFileName`

`Mesh.Color.Eighteen`
: Color 18 in color carousel
  Default value: `{0,255,52}`
  Saved in: `General.OptionsFileName`

`Mesh.Color.Nineteen`
: Color 19 in color carousel
  Default value: `{184,0,255}`
  Saved in: `General.OptionsFileName`

B.4 Solver options list

`Solver.Executable0`
: System command to launch solver 0
  Default value: `""`
  Saved in: `General.SessionFileName`

`Solver.Executable1`
: System command to launch solver 1
  Default value: `""`
  Saved in: `General.SessionFileName`

`Solver.Executable2`
: System command to launch solver 2
  Default value: `""`
  Saved in: `General.SessionFileName`

`Solver.Executable3`
: System command to launch solver 3
  Default value: `""`
  Saved in: `General.SessionFileName`

`Solver.Executable4`
: System command to launch solver 4
  Default value: `""`
  Saved in: `General.SessionFileName`

`Solver.Name0`
: Name of solver 0
  Default value: `"GetDP"`
  Saved in: `General.SessionFileName`

`Solver.Name1`
: Name of solver 1
Default value: `""`
Saved in: `General.SessionFileName`

`Solver.Name2`
: Name of solver 2
Default value: `""`
Saved in: `General.SessionFileName`

`Solver.Name3`
: Name of solver 3
Default value: `""`
Saved in: `General.SessionFileName`

`Solver.Name4`
: Name of solver 4
Default value: `""`
Saved in: `General.SessionFileName`

`Solver.RemoteLogin0`
: Command to login to a remote host to launch solver 0
Default value: `""`
Saved in: `General.SessionFileName`

`Solver.RemoteLogin1`
: Command to login to a remote host to launch solver 1
Default value: `""`
Saved in: `General.SessionFileName`

`Solver.RemoteLogin2`
: Command to login to a remote host to launch solver 2
Default value: `""`
Saved in: `General.SessionFileName`

`Solver.RemoteLogin3`
: Command to login to a remote host to launch solver 3
Default value: `""`
Saved in: `General.SessionFileName`

`Solver.RemoteLogin4`
: Command to login to a remote host to launch solver 4
Default value: `""`
Saved in: `General.SessionFileName`

`Solver.SocketName`
: Base name of socket (TCP/IP if it contains the ':' character, UNIX otherwise)
Default value: `".gmshsock"`
Saved in: `General.OptionsFileName`

`Solver.AlwaysListen`
: Always listen to incoming connection requests?
Default value: `0`
Saved in: `General.OptionsFileName`

`Solver.AutoArchiveOutputFiles`
: Automatically archive output files after each computation
Default value: `0`
Saved in: `General.OptionsFileName`

`Solver.AutoCheck`
: Automatically check model every time a parameter is changed
Default value: `1`
Saved in: `General.OptionsFileName`

`Solver.AutoSaveDatabase`
: Automatically save database after each computation
Default value: `0`
Saved in: `General.OptionsFileName`

`Solver.AutoMesh`
: Automatically mesh if necesssary
Default value: `1`
Saved in: `General.OptionsFileName`

`Solver.AutoMergeFile`
: Automatically merge result files
Default value: `1`
Saved in: `General.OptionsFileName`

`Solver.AutoHideNewViews`
: Automcatically hide newly merged results
Default value: `0`
Saved in: `General.OptionsFileName`

`Solver.AutoShowLastStep`
: Automatically show the last time step in newly merged results
Default value: `1`
Saved in: `General.OptionsFileName`

`Solver.Plugins`
: Enable default solver plugins?
Default value: `0`
Saved in: `General.OptionsFileName`

`Solver.Timeout`
: Time (in seconds) before closing the socket if no connection is happening
Default value: `5`
Saved in: `General.OptionsFileName`

B.5 Post-processing options list

`PostProcessing.AnimationDelay`
Delay (in seconds) between frames in automatic animation mode
Default value: `0.1`
Saved in: `General.OptionsFileName`

`PostProcessing.AnimationCycle`
Cycle through time steps (0) or views (1) for animations
Default value: `0`
Saved in: `General.OptionsFileName`

`PostProcessing.AnimationStep`
Step increment for animations
Default value: `1`
Saved in: `General.OptionsFileName`

`PostProcessing.CombineRemoveOriginal`
Remove original views after a Combine operation
Default value: `1`
Saved in: `General.OptionsFileName`

`PostProcessing.Format`
Default file format for post-processing views (0=ASCII view, 1=binary view, 2=parsed view, 3=STL triangulation, 4=raw text, 5=Gmsh mesh, 6=MED file, 10=automatic)
Default value: `10`
Saved in: `General.OptionsFileName`

`PostProcessing.HorizontalScales`
Display value scales horizontally
Default value: `1`
Saved in: `General.OptionsFileName`

`PostProcessing.Link`
Link post-processing views (0=none, 1/2=changes in visible/all, 3/4=everything in visible/all)
Default value: `0`
Saved in: `General.OptionsFileName`

`PostProcessing.NbViews`
Current number of views merged (read-only)
Default value: `0`
Saved in: `-`

`PostProcessing.Plugins`
Enable default post-processing plugins?
Default value: `1`
Saved in: `General.OptionsFileName`

`PostProcessing.Smoothing`
: Apply (non-reversible) smoothing to post-processing view when merged
  Default value: `0`
  Saved in: `General.OptionsFileName`

`View.Attributes`
: Optional string attributes
  Default value: `""`
  Saved in: `General.OptionsFileName`

`View.AxesFormatX`
: Number format for X-axis (in standard C form)
  Default value: `"%.3g"`
  Saved in: `General.OptionsFileName`

`View.AxesFormatY`
: Number format for Y-axis (in standard C form)
  Default value: `"%.3g"`
  Saved in: `General.OptionsFileName`

`View.AxesFormatZ`
: Number format for Z-axis (in standard C form)
  Default value: `"%.3g"`
  Saved in: `General.OptionsFileName`

`View.AxesLabelX`
: X-axis label
  Default value: `""`
  Saved in: `General.OptionsFileName`

`View.AxesLabelY`
: Y-axis label
  Default value: `""`
  Saved in: `General.OptionsFileName`

`View.AxesLabelZ`
: Z-axis label
  Default value: `""`
  Saved in: `General.OptionsFileName`

`View.FileName`
: Default post-processing view file name
  Default value: `""`
  Saved in: `-`

`View.Format`
: Number format (in standard C form)
  Default value: `"%.3g"`
  Saved in: `General.OptionsFileName`

`View.GeneralizedRaiseX`
Generalized elevation of the view along X-axis (in model coordinates)
Default value: `"v0"`
Saved in: `General.OptionsFileName`

`View.GeneralizedRaiseY`
Generalized elevation of the view along Y-axis (in model coordinates)
Default value: `"v1"`
Saved in: `General.OptionsFileName`

`View.GeneralizedRaiseZ`
Generalized elevation of the view along Z-axis (in model coordinates)
Default value: `"v2"`
Saved in: `General.OptionsFileName`

`View.Name`
Default post-processing view name
Default value: `""`
Saved in: `-`

`View.Stipple0`
First stippling pattern
Default value: `"1*0x1F1F"`
Saved in: `General.OptionsFileName`

`View.Stipple1`
Second stippling pattern
Default value: `"1*0x3333"`
Saved in: `General.OptionsFileName`

`View.Stipple2`
Third stippling pattern
Default value: `"1*0x087F"`
Saved in: `General.OptionsFileName`

`View.Stipple3`
Fourth stippling pattern
Default value: `"1*0xCCCF"`
Saved in: `General.OptionsFileName`

`View.Stipple4`
Fifth stippling pattern
Default value: `"2*0x1111"`
Saved in: `General.OptionsFileName`

`View.Stipple5`
Sixth stippling pattern
Default value: `"2*0x0F0F"`
Saved in: `General.OptionsFileName`

`View.Stipple6`
: Seventh stippling pattern
  Default value: `"1*0xCFFF"`
  Saved in: `General.OptionsFileName`

`View.Stipple7`
: Eighth stippling pattern
  Default value: `"2*0x0202"`
  Saved in: `General.OptionsFileName`

`View.Stipple8`
: Ninth stippling pattern
  Default value: `"2*0x087F"`
  Saved in: `General.OptionsFileName`

`View.Stipple9`
: Tenth stippling pattern
  Default value: `"1*0xFFFF"`
  Saved in: `General.OptionsFileName`

`View.AbscissaRangeType`
: Ascissa scale range type (1=default, 2=custom)
  Default value: `1`
  Saved in: `General.OptionsFileName`

`View.AdaptVisualizationGrid`
: Use adaptive visualization grid (for high-order elements)?
  Default value: `0`
  Saved in: `General.OptionsFileName`

`View.AngleSmoothNormals`
: Threshold angle below which normals are not smoothed
  Default value: `30`
  Saved in: `General.OptionsFileName`

`View.ArrowSizeMax`
: Maximum display size of arrows (in pixels)
  Default value: `60`
  Saved in: `General.OptionsFileName`

`View.ArrowSizeMin`
: Minimum display size of arrows (in pixels)
  Default value: `0`
  Saved in: `General.OptionsFileName`

`View.AutoPosition`
: Position the scale or 2D plot automatically (0: manual, 1: automatic, 2: top left, 3: top right, 4: bottom left, 5: bottom right, 6: top, 7: bottom, 8: left, 9: right, 10: full, 11: top third)
  Default value: `1`
  Saved in: `General.OptionsFileName`

`View.Axes`
: Axes (0=none, 1=simple axes, 2=box, 3=full grid, 4=open grid, 5=ruler)
Default value: `0`
Saved in: `General.OptionsFileName`

`View.AxesMikado`
: Mikado axes style
Default value: `0`
Saved in: `General.OptionsFileName`

`View.AxesAutoPosition`
: Position the axes automatically
Default value: `1`
Saved in: `General.OptionsFileName`

`View.AxesMaxX`
: Maximum X-axis coordinate
Default value: `1`
Saved in: `General.OptionsFileName`

`View.AxesMaxY`
: Maximum Y-axis coordinate
Default value: `1`
Saved in: `General.OptionsFileName`

`View.AxesMaxZ`
: Maximum Z-axis coordinate
Default value: `1`
Saved in: `General.OptionsFileName`

`View.AxesMinX`
: Minimum X-axis coordinate
Default value: `0`
Saved in: `General.OptionsFileName`

`View.AxesMinY`
: Minimum Y-axis coordinate
Default value: `0`
Saved in: `General.OptionsFileName`

`View.AxesMinZ`
: Minimum Z-axis coordinate
Default value: `0`
Saved in: `General.OptionsFileName`

`View.AxesTicsX`
: Number of tics on the X-axis
Default value: `5`
Saved in: `General.OptionsFileName`

`View.AxesTicsY`
: Number of tics on the Y-axis
Default value: `5`
Saved in: `General.OptionsFileName`

`View.AxesTicsZ`
: Number of tics on the Z-axis
Default value: `5`
Saved in: `General.OptionsFileName`

`View.Boundary`
: Draw the 'N minus b'-dimensional boundary of the element (N=element dimension, b=option value)
Default value: `0`
Saved in: `General.OptionsFileName`

`View.CenterGlyphs`
: Center glyphs (arrows, numbers, etc.)? (0=left, 1=centered, 2=right)
Default value: `0`
Saved in: `General.OptionsFileName`

`View.Clip`
: Enable clipping planes? (Plane[i]=2^i, i=0,...,5)
Default value: `0`
Saved in: `-`

`View.ColormapAlpha`
: Colormap alpha channel value (used only if != 1)
Default value: `1`
Saved in: `General.OptionsFileName`

`View.ColormapAlphaPower`
: Colormap alpha channel power
Default value: `0`
Saved in: `General.OptionsFileName`

`View.ColormapBeta`
: Colormap beta parameter (gamma = 1-beta)
Default value: `0`
Saved in: `General.OptionsFileName`

`View.ColormapBias`
: Colormap bias
Default value: `0`
Saved in: `General.OptionsFileName`

`View.ColormapCurvature`
: Colormap curvature or slope coefficient
Default value: `0`
Saved in: `General.OptionsFileName`

`View.ColormapInvert`
: Invert the color values, i.e., replace x with (255-x) in the colormap?
Default value: `0`
Saved in: `General.OptionsFileName`

`View.ColormapNumber`
: Default colormap number
Default value: `2`
Saved in: `General.OptionsFileName`

`View.ColormapRotation`
: Incremental colormap rotation
Default value: `0`
Saved in: `General.OptionsFileName`

`View.ColormapSwap`
: Swap the min/max values in the colormap?
Default value: `0`
Saved in: `General.OptionsFileName`

`View.ComponentMap0`
: Forced component 0 (if View.ForceComponents > 0)
Default value: `0`
Saved in: `General.OptionsFileName`

`View.ComponentMap1`
: Forced component 1 (if View.ForceComponents > 0)
Default value: `1`
Saved in: `General.OptionsFileName`

`View.ComponentMap2`
: Forced component 2 (if View.ForceComponents > 0)
Default value: `2`
Saved in: `General.OptionsFileName`

`View.ComponentMap3`
: Forced component 3 (if View.ForceComponents > 0)
Default value: `3`
Saved in: `General.OptionsFileName`

`View.ComponentMap4`
: Forced component 4 (if View.ForceComponents > 0)
Default value: `4`
Saved in: `General.OptionsFileName`

`View.ComponentMap5`
: Forced component 5 (if View.ForceComponents > 0)
Default value: `5`
Saved in: `General.OptionsFileName`

`View.ComponentMap6`
Forced component 6 (if View.ForceComponents > 0)
Default value: `6`
Saved in: `General.OptionsFileName`

`View.ComponentMap7`
Forced component 7 (if View.ForceComponents > 0)
Default value: `7`
Saved in: `General.OptionsFileName`

`View.ComponentMap8`
Forced component 8 (if View.ForceComponents > 0)
Default value: `8`
Saved in: `General.OptionsFileName`

`View.CustomAbscissaMax`
User-defined maximum abscissa value
Default value: `0`
Saved in: `-`

`View.CustomAbscissaMin`
User-defined minimum abscissa value
Default value: `0`
Saved in: `-`

`View.CustomMax`
User-defined maximum value to be displayed
Default value: `0`
Saved in: `-`

`View.CustomMin`
User-defined minimum value to be displayed
Default value: `0`
Saved in: `-`

`View.DisplacementFactor`
Displacement amplification
Default value: `1`
Saved in: `General.OptionsFileName`

`View.DrawHexahedra`
Display post-processing hexahedra?
Default value: `1`
Saved in: `General.OptionsFileName`

`View.DrawLines`
Display post-processing lines?
Default value: `1`
Saved in: `General.OptionsFileName`

`View.DrawPoints`
: Display post-processing points?
Default value: `1`
Saved in: `General.OptionsFileName`

`View.DrawPrisms`
: Display post-processing prisms?
Default value: `1`
Saved in: `General.OptionsFileName`

`View.DrawPyramids`
: Display post-processing pyramids?
Default value: `1`
Saved in: `General.OptionsFileName`

`View.DrawQuadrangles`
: Display post-processing quadrangles?
Default value: `1`
Saved in: `General.OptionsFileName`

`View.DrawScalars`
: Display scalar values?
Default value: `1`
Saved in: `General.OptionsFileName`

`View.DrawSkinOnly`
: Draw only the skin of 3D scalar views?
Default value: `0`
Saved in: `General.OptionsFileName`

`View.DrawStrings`
: Display post-processing annotation strings?
Default value: `1`
Saved in: `General.OptionsFileName`

`View.DrawTensors`
: Display tensor values?
Default value: `1`
Saved in: `General.OptionsFileName`

`View.DrawTetrahedra`
: Display post-processing tetrahedra?
Default value: `1`
Saved in: `General.OptionsFileName`

`View.DrawTriangles`
: Display post-processing triangles?
Default value: `1`
Saved in: `General.OptionsFileName`

`View.DrawVectors`
Display vector values?
Default value: `1`
Saved in: `General.OptionsFileName`

`View.Explode`
Element shrinking factor (between 0 and 1)
Default value: `1`
Saved in: `General.OptionsFileName`

`View.ExternalView`
Index of the view used to color vector fields (-1=self)
Default value: `-1`
Saved in: `General.OptionsFileName`

`View.FakeTransparency`
Use fake transparency (cheaper than the real thing, but incorrect)
Default value: `0`
Saved in: `General.OptionsFileName`

`View.ForceNumComponents`
Force number of components to display (see View.ComponentMapN for mapping)
Default value: `0`
Saved in: `General.OptionsFileName`

`View.GeneralizedRaiseFactor`
Generalized raise amplification factor
Default value: `1`
Saved in: `General.OptionsFileName`

`View.GeneralizedRaiseView`
Index of the view used for generalized raise (-1=self)
Default value: `-1`
Saved in: `General.OptionsFileName`

`View.GlyphLocation`
Glyph (arrow, number, etc.) location (1=center of gravity, 2=node)
Default value: `1`
Saved in: `General.OptionsFileName`

`View.Height`
Height (in pixels) of the scale or 2D plot
Default value: `200`
Saved in: `General.OptionsFileName`

`View.IntervalsType`
Type of interval display (1=iso, 2=continuous, 3=discrete, 4=numeric)
Default value: `2`
Saved in: `General.OptionsFileName`

`View.Light`
: Enable lighting for the view
Default value: `1`
Saved in: `General.OptionsFileName`

`View.LightLines`
: Light element edges
Default value: `1`
Saved in: `General.OptionsFileName`

`View.LightTwoSide`
: Light both sides of surfaces (leads to slower rendering)
Default value: `1`
Saved in: `General.OptionsFileName`

`View.LineType`
: Display lines as solid color segments (0) or 3D cylinders (1)
Default value: `0`
Saved in: `General.OptionsFileName`

`View.LineWidth`
: Display width of lines (in pixels)
Default value: `1`
Saved in: `General.OptionsFileName`

`View.MaxRecursionLevel`
: Maximum recursion level for adaptive views
Default value: `0`
Saved in: `General.OptionsFileName`

`View.Max`
: Maximum value in the view (read-only)
Default value: `0`
Saved in: `-`

`View.MaxX`
: Maximum view coordinate along the X-axis (read-only)
Default value: `0`
Saved in: `-`

`View.MaxY`
: Maximum view coordinate along the Y-axis (read-only)
Default value: `0`
Saved in: `-`

`View.MaxZ`
: Maximum view coordinate along the Z-axis (read-only)
Default value: `0`
Saved in: `-`

`View.Min`
: Minimum value in the view (read-only)
Default value: `0`
Saved in: `-`

`View.MinX`
: Minimum view coordinate along the X-axis (read-only)
Default value: `0`
Saved in: `-`

`View.MinY`
: Minimum view coordinate along the Y-axis (read-only)
Default value: `0`
Saved in: `-`

`View.MinZ`
: Minimum view coordinate along the Z-axis (read-only)
Default value: `0`
Saved in: `-`

`View.NbIso`
: Number of intervals
Default value: `10`
Saved in: `General.OptionsFileName`

`View.NbTimeStep`
: Number of time steps in the view (do not change this!)
Default value: `1`
Saved in: `-`

`View.NormalRaise`
: Elevation of the view along the normal (in model coordinates)
Default value: `0`
Saved in: `-`

`View.Normals`
: Display size of normal vectors (in pixels)
Default value: `0`
Saved in: `General.OptionsFileName`

`View.OffsetX`
: Translation of the view along X-axis (in model coordinates)
Default value: `0`
Saved in: `-`

`View.OffsetY`
: Translation of the view along Y-axis (in model coordinates)
Default value: `0`
Saved in: `-`

`View.OffsetZ`
: Translation of the view along Z-axis (in model coordinates)
Default value: `0`
Saved in: `-`

`View.PointSize`
: Display size of points (in pixels)
Default value: `3`
Saved in: `General.OptionsFileName`

`View.PointType`
: Display points as solid color dots (0), 3D spheres (1), scaled dots (2) or scaled spheres (3)
Default value: `0`
Saved in: `General.OptionsFileName`

`View.PositionX`
: X position (in pixels) of the scale or 2D plot (< 0: measure from right edge; >= 1e5: centered)
Default value: `100`
Saved in: `General.OptionsFileName`

`View.PositionY`
: Y position (in pixels) of the scale or 2D plot (< 0: measure from bottom edge; >= 1e5: centered)
Default value: `50`
Saved in: `General.OptionsFileName`

`View.RaiseX`
: Elevation of the view along X-axis (in model coordinates)
Default value: `0`
Saved in: `-`

`View.RaiseY`
: Elevation of the view along Y-axis (in model coordinates)
Default value: `0`
Saved in: `-`

`View.RaiseZ`
: Elevation of the view along Z-axis (in model coordinates)
Default value: `0`
Saved in: `-`

`View.RangeType`
: Value scale range type (1=default, 2=custom, 3=per time step)
Default value: `1`
Saved in: `General.OptionsFileName`

`View.Sampling`
: Element sampling rate (draw one out every 'Sampling' elements)
Default value: `1`
Saved in: `General.OptionsFileName`

`View.SaturateValues`
: Saturate the view values to custom min and max (1=true, 0=false)
Default value: `0`
Saved in: `General.OptionsFileName`

`View.ScaleType`
: Value scale type (1=linear, 2=logarithmic, 3=double logarithmic)
  Default value: `1`
  Saved in: `General.OptionsFileName`

`View.ShowElement`
: Show element boundaries?
  Default value: `0`
  Saved in: `General.OptionsFileName`

`View.ShowScale`
: Show value scale?
  Default value: `1`
  Saved in: `General.OptionsFileName`

`View.ShowTime`
: Time display mode (0=hidden, 1=time value if multi-step, 2=time value always, 3=time step if multi-step, 4=time step always)
  Default value: `3`
  Saved in: `General.OptionsFileName`

`View.SmoothNormals`
: Smooth the normals?
  Default value: `0`
  Saved in: `General.OptionsFileName`

`View.Stipple`
: Stipple curves in 2D plots?
  Default value: `0`
  Saved in: `General.OptionsFileName`

`View.Tangents`
: Display size of tangent vectors (in pixels)
  Default value: `0`
  Saved in: `General.OptionsFileName`

`View.TargetError`
: Target representation error for adaptive views
  Default value: `0.01`
  Saved in: `General.OptionsFileName`

`View.TensorType`
: Tensor Visualization Type
  Default value: `1`
  Saved in: `General.OptionsFileName`

`View.TimeStep`
: Current time step displayed
  Default value: `0`
  Saved in: `-`

`View.TransformXX`
: Element (1,1) of the 3x3 coordinate transformation matrix
  Default value: `1`
  Saved in: `-`

`View.TransformXY`
: Element (1,2) of the 3x3 coordinate transformation matrix
  Default value: `0`
  Saved in: `-`

`View.TransformXZ`
: Element (1,3) of the 3x3 coordinate transformation matrix
  Default value: `0`
  Saved in: `-`

`View.TransformYX`
: Element (2,1) of the 3x3 coordinate transformation matrix
  Default value: `0`
  Saved in: `-`

`View.TransformYY`
: Element (2,2) of the 3x3 coordinate transformation matrix
  Default value: `1`
  Saved in: `-`

`View.TransformYZ`
: Element (2,3) of the 3x3 coordinate transformation matrix
  Default value: `0`
  Saved in: `-`

`View.TransformZX`
: Element (3,1) of the 3x3 coordinate transformation matrix
  Default value: `0`
  Saved in: `-`

`View.TransformZY`
: Element (3,2) of the 3x3 coordinate transformation matrix
  Default value: `0`
  Saved in: `-`

`View.TransformZZ`
: Element (3,3) of the 3x3 coordinate transformation matrix
  Default value: `1`
  Saved in: `-`

`View.Type`
: Type of plot (1=3D, 2=2D space, 3=2D time)
  Default value: `1`
  Saved in: `-`

`View.UseGeneralizedRaise`
: Use generalized raise?
Default value: `0`
Saved in: `General.OptionsFileName`

`View.VectorType`
: Vector display type (1=segment, 2=arrow, 3=pyramid, 4=3D arrow, 5=displacement, 6=comet)
Default value: `4`
Saved in: `General.OptionsFileName`

`View.Visible`
: Is the view visible?
Default value: `1`
Saved in: `-`

`View.Width`
: Width (in pixels) of the scale or 2D plot
Default value: `300`
Saved in: `General.OptionsFileName`

`View.Color.Points`
: Point color
Default value: `{0,0,0}`
Saved in: `General.OptionsFileName`

`View.Color.Lines`
: Line color
Default value: `{0,0,0}`
Saved in: `General.OptionsFileName`

`View.Color.Triangles`
: Triangle color
Default value: `{0,0,0}`
Saved in: `General.OptionsFileName`

`View.Color.Quadrangles`
: Quadrangle color
Default value: `{0,0,0}`
Saved in: `General.OptionsFileName`

`View.Color.Tetrahedra`
: Tetrahedron color
Default value: `{0,0,0}`
Saved in: `General.OptionsFileName`

`View.Color.Hexahedra`
: Hexahedron color
Default value: `{0,0,0}`
Saved in: `General.OptionsFileName`

`View.Color.Prisms`
: Prism color
Default value: `{0,0,0}`
Saved in: `General.OptionsFileName`

`View.Color.Pyramids`
: Pyramid color
Default value: `{0,0,0}`
Saved in: `General.OptionsFileName`

`View.Color.Tangents`
: Tangent vector color
Default value: `{255,255,0}`
Saved in: `General.OptionsFileName`

`View.Color.Normals`
: Normal vector color
Default value: `{255,0,0}`
Saved in: `General.OptionsFileName`

`View.Color.Text2D`
: 2D text color
Default value: `{0,0,0}`
Saved in: `General.OptionsFileName`

`View.Color.Text3D`
: 3D text color
Default value: `{0,0,0}`
Saved in: `General.OptionsFileName`

`View.Color.Axes`
: Axes color
Default value: `{0,0,0}`
Saved in: `General.OptionsFileName`

`View.Color.Background2D`
: Bacground color for 2D plots
Default value: `{255,255,255}`
Saved in: `General.OptionsFileName`

`View.ColorTable`
: Color table used to draw the view
Saved in: `General.OptionsFileName`

Appendix C Information for developers

Gmsh is written in C++, the scripting language is parsed using Lex and Yacc (actually, Flex and Bison), and the GUI relies on OpenGL for the 3D graphics and FLTK (`http://www.fltk.org`) for the widget set. Gmsh's build system is based on CMake (`http://www.cmake.org`). Practical notes on how to compile Gmsh's source code are included in the distribution. See Appendix D [Frequently asked questions], page 201, for more information.

C.1 Getting the source

Stable releases and nightly source snapshots are available from `http://geuz.org/gmsh/src/`.

You can also access the subversion repository directly:

1. The first time you want to download the latest full source, type:

   ```
   svn co https://geuz.org/svn/gmsh/trunk gmsh
   ```

   You will be asked to accept the security certificate and to provide your username and password. (Use gmsh/gmsh for read-only access.)
2. To update your local version to the latest and greatest, go in the gmsh directory and type:

   ```
   svn update
   ```
3. If you have write access, to commit your changes to the central repository, go in the gmsh directory and type:

   ```
   svn commit
   ```

C.2 Source code structure

Gmsh's code is structured in several subdirectories, roughly separated between the four core modules (`Geo`, `Mesh`, `Solver`, `Post`) and associated utilities (`Common`, `Numeric`) on one hand, and the graphics (`Graphics`) and interface (`Fltk`, `Parser`) code on the other.

The geometry and mesh modules are based on an object-oriented model class (`Geo/GModel.h`), built upon abstract geometrical entity classes (`Geo/GVertex.h`, `Geo/GEdge.h`, `Geo/GFace.h` and `Geo/GRegion.h`). Concrete implementations of the geometrical entity classes are provided for each supported CAD kernel (e.g. `Geo/gmshVertex.h` for geometry points in Gmsh's native CAD format, or `Geo/OCCVertex.h` for geometry points from OpenCASCADE). The post-processing module is based on the concept of views (`Post/PView.h`) and abstract data containers (derived from `Post/PViewData.h`).

C.3 Coding style

If you plan to contribute code to the Gmsh project, here are some easy rules to make the code easy to read/debug/maintain:

1. Please enable full warnings for your compiler (e.g. `-Wall` with `g++`) and don't commit until there is no warning left.
2. Use memory checking tools to detect memory leaks and other nasty memory problems. For example, you can use

- Valgrind on Linux:

  ```
  valgrind --leak-check=yes --show-reachable=yes gmsh file.geo -3
  ```

- GMALLOC on Mac OS X:

  ```
  (gdb) set env DYLD_INSERT_LIBRARIES /usr/lib/libgmalloc.dylib
  ```

- LIBNJAMD:

  ```
  export LD_PRELOAD=libnjamd.so
  kill -USR1
  ```

- Purify
- Memprof
- . . .

3. always use the `Msg::` class to print information or errors
4. indent your files (2 spaces) and convert all tabs to spaces
5. follow the style used in the existing code when adding something new (spaces after commas, opening braces for functions on a separate line, opening braces for loops and tests on the same line, etc.)

C.4 Adding a new option

To add a new option in Gmsh:

1. create the option in the `CTX` class (‘`Common/Context.h`’) if it’s a classical option, or in the `PViewOptions` class (‘`Post/PViewOptions.h`’) if it’s a post-processing view-dependent option;
2. in ‘`Common/DefaultOptions.h`’, give a name (for the parser to be able to access it), a reference to a handling routine (i.e. `opt_XXX`) and a default value for this option;
3. create the handling routine `opt_XXX` in ‘`Common/Options.cpp`’ (and add the prototype in ‘`Common/Options.h`’);
4. optional: create the associated widget in ‘`Fltk/optionWindow.cpp`’;

Appendix D Frequently asked questions

D.1 The basics

1. What is Gmsh?

   Gmsh is an automatic three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. With Gmsh you can create or import 1D, 2D and 3D geometrical models, mesh them, launch external finite element solvers and visualize solutions. Gmsh can be used either as a stand-alone program (graphical or not) or as a C++ library.

2. What are the terms and conditions of use?

   Gmsh is distributed under the terms of the GNU General Public License, with an exception to allow for easier linking with external libraries. See Appendix G [License], page 225 for more information.

3. What does 'Gmsh' mean?

   Nothing... The name was derived from a previous version called "msh" (a shortcut for "mesh"), with the "g" prefix added to differentiate it. The default mesh file format used by Gmsh still uses the '`.msh`' extension.

   In English people tend to pronounce 'Gmsh' as "gee-mesh".

4. Where can I find more information?

   `http://geuz.org/gmsh` is the primary location to obtain information about Gmsh. There you will for example find the complete reference manual, a bug tracking database and a searchable archive of the Gmsh mailing list (`gmsh@geuz.org`).

D.2 Installation problems

1. Which OSes does Gmsh run on?

   Gmsh runs on Windows, Mac OS X, Linux and most Unix variants.

2. Are there additional requirements to run Gmsh?

   You should have the OpenGL libraries installed on your system, and in the path of the library loader. A free replacement for OpenGL can be found at `http://www.mesa3d.org`.

3. How do I compile Gmsh from the source code?

   You need cmake (`http://www.cmake.org`) and a C++ compiler. See the '`README.txt`' file in the top-level source directory for more information.

4. Where does Gmsh save its configuration files?

   Gmsh will attempt to save temporary files and persistent configuration options first in the `$GMSH_HOME` directory, then in `$APPDATA` (on Windows) or `$HOME` (on other OSes), then in `$TMP`, and finally in `$TEMP`, in that order. If none of these variables are defined, Gmsh will try to save/load its configuration files from the current working directory.

D.3 General questions

1. Gmsh (from a binary distribution) complains about missing libraries.

   On Windows, if your system complains about missing '`OPENGL32.DLL`' or '`GLU32.DLL`' libraries, then OpenGL is not properly installed on your machine. You can download OpenGL from Microsoft's web site, or directly from `http://www.opengl.org`.

   On Unix try 'ldd gmsh' (or 'otool -L gmsh' on Mac OS X) to check if all the required shared libraries are installed on your system. If not, install them. If it still doesn't work, recompile Gmsh from the source code.
2. Gmsh keeps re-displaying its graphics when other windows partially hide the graphical window.

   Disable opaque move in your window manager.
3. The graphics display very slowly.

   Are you are executing Gmsh from a remote host (via the network) without GLX? You should turn double buffering off (with the '-nodb' command line option).
4. There is an ugly "ghost triangulation" in the vector PostScript/PDF files generated by Gmsh!

   No, there isn't. This "ghost triangulation" is due to the fact that most PostScript previewers nowadays antialias the graphic primitives when they display the page on screen. (For example, in gv, you can disable antialising with the 'State->Antialias' menu.) You should not see this ghost triangulation in the printed output (on paper).
5. How can I save GIF, JPEG, ..., images?

   Just choose the appropriate format in 'File->Save As'. By default Gmsh guesses the format from the file extension, so you can just type '`myfile.jpg`' in the dialog and Gmsh will automatically create a JPEG image file.
6. How can I save MPEG, AVI, ..., animations?

   Using a script. Have a look at '`tutorial/t8.geo`' or '`demos/anim.script`' for some examples.
7. Can I change values in input fields with the mouse in the GUI?

   Yes: dragging the mouse in a numeric input field slides the value! The left button moves one step per pixel, the middle by '10*step', and the right button by '100*step'.
8. Can I copy messages to the clipboard?

   Yes: selecting the content of an input field, or lines in the message console ('Tools->Message Console'), copies the selected text to the clipboard.

D.4 Geometry module

1. Does Gmsh support NURBS curves/surfaces?

   Yes, but only via STEP, IGES or BREP model import (not in '`.geo`' files). Gmsh has to be compiled with OpenCASCADE support for this to work.
2. Gmsh is very slow when I use many transformations (Translate, Rotate, Symmetry, Extrude, etc.). What's wrong?

   The default behavior of Gmsh is to check and suppress all duplicate entities (points, lines and surfaces) each time a transformation command is issued. This can slow down things a lot if many transformations are performed. There are two solutions to this problem:

- you may save the unrolled geometry in another file (e.g. with gmsh file.geo -0), and use this new file for subsequent computations;
- or you may set the `Geometry.AutoCoherence` option to 0. This will prevent any automatic duplicate check/replacement. If you still need to remove the duplicates entities, simply add `Coherence;` at strategic locations in your geo files (e.g. before the creation of line loops, etc.).

3. How can I display only selected parts of my model?

   Use 'Tools->Visibility'. This allows you to select elementary entities and physical groups, as well as mesh elements, in a variety of ways (in a list or tree browser, by identification number, interactively, or per window).

4. Can I edit STEP/IGES/BRep models?

   Not yet. At the moment you can only change mesh element sizes, define volumes and physical groups, or delete entities. The easiest way to do this is to merge the model in a '`.geo`' file using `Merge "file.step";` and add the relevant scripting command after that. We plan to add more advanced editing features in the future (to delete entities, to create "mixed" surfaces and volumes, to export in '`.geo`' format, etc.).

5. How can I build modular geometries?

   Define common geometrical objects and options in separate files, reusable in all your problem definition structures. Then Include the files in your main project file.

D.5 Mesh module

1. What should I do when the 2D unstructured algorithm fails?

   Verify that the curves in the model do not self-intersect. If 'Mesh.RandomFactor*size(triangle)/size(model) approaches machine accuracy, increase Mesh.RandomFactor.

   If everything fails file a bug report with the version of your operating system and the full geometry.

2. What should I do when the 3D unstructured algorithm fails?

   Verify that the surfaces in your model do not self-intersect or partially overlap. If they don't, try the other 3D algorithms ('Tool->Options->Mesh->General->3D algorithm') or try to adapt the mesh element sizes in your input file so that the surface mesh better matches the geometrical details of the model.

   If nothing works, file a bug report with the version of your operating system and and the full geometry.

3. My 2D meshes of IGES files present gaps between surfaces

   IGES files do not contain the topology of the model, and tolerance problems can thus appear when the OpenCASCADE importer cannot identify two (close) curves as actually being identical.

   The best solution is to *not use IGES and use STEP* instead. If you really have to use IGES, check that you don't have duplicate curves (e.g. by displaying their numbers in the GUI with 'Tools->Options->Geometry->Visibility->Line numbers'). If there are duplicates, try to change the geometrical tolerance and sew the faces (see options in 'Tools->Options->Geometry->General').

4. The quality of the elements generated by the 3D algorithm is very bad.

   Use 'Optimize quality' in the mesh menu.

5. Non-recombined 3D extruded meshes sometimes fail.

   The swapping algorithm is not very clever at the moment. Try to change the surface mesh a bit, or recombine your mesh to generate prisms or hexahedra instead of tetrahedra.

6. Does Gmsh automatically couple unstructured tetrahedral meshes and structured hexahedral meshed using pyramids?

   No. We need your help to implement this.

7. Can I explicitly assign region numbers to extruded layers?

   No, this feature has been removed in Gmsh 2.0. You must use the standard entity number instead.

8. Did you remove the elliptic mesh generator in Gmsh 2.0?

   Yes. You can achieve the same result by using the transfinite algorithm with smoothing (e.g., with `Mesh.Smoothing = 10`).

9. Does Gmsh support curved elements?

   Yes, Gmsh can generate both 1st order and 2nd order elements. To generate second order elements, click on 'High order' in the mesh menu after the mesh is completed. To always generate 2nd order elements, select 'Generate second order elements' in the mesh option panel. From the command line, you can also use `-order 2`.

10. Can I import an existing surface mesh in Gmsh and use it to build a 3D mesh?

    Yes, you can import a surface mesh in any one of the supported mesh file formats, define a volume, and mesh it. For an example see '`demos/sphere-discrete.geo`'.

11. How do I define boundary conditions or material properties in Gmsh?

    By design, Gmsh does not try to incorporate every possible definition of boundary conditions or material properties—this is a job best left to the solver. Instead, Gmsh provides a simple mechanism to tag groups of elements, and it is up to the solver to interpret these tags as boundary conditions, materials, etc. Associating tags with elements in Gmsh is done by defining Physical entities (Physical Points, Physical Lines, Physical Surfaces and Physical Volumes). See the reference manual as well as the tutorials (in particular '`tutorial/t1.geo`') for a detailed description and some examples.

12. How can I display only the mesh associated with selected geometrical entities?

    See "How can I display only selected parts of my model?".

13. How can I "explore" a mesh (for example, to see inside a complex structure)?

    You can use 'Tools->Clipping Planes' to clip the region of interest. You can define up to 6 clipping planes in Gmsh (i.e., enough to define a "cube" inside your model) and each plane can clip either the geometry, the mesh, the post-processing views, or any combination of the above. The clipping planes are defined using the four coefficients A,B,C,D of the equation A*x+B*y+C*y+D=0, which can be adjusted interactively by dragging the mouse in the input fields.

14. What is the signification of Rho, Eta and Gamma in Tools->Statistics?

    They measure the quality of the tetrahedra in a mesh:

    Gamma ~ inscribed_radius / circumscribed_radius Eta ~ volume^(2/3) / sum_edge_length^2 Rho ~ min_edge_length / max_edge_length

    For the exact definitions, see Geo/MElement.cpp. The graphs plot the the number of elements vs. the quality measure.

15. Why don't the vertex and/or elememt numbers on the screen match the numbers in the mesh file?

    Gmsh reindexes the mesh vertices and elements so that they are numbered in a continuous sequence in the output files. The numbers displayed on screen after mesh generation thus usually differ from the ones saved in the mesh files. To check the actual numbers saved in the output file just load the mesh file back using 'File->Open'.

D.6 Solver module

1. How do I integrate my own solver with Gmsh?

   Gmsh uses the ONELAB interface (`http://www.onelab.info`) to interact with external solvers. Have a look at the GetDP finite element solver (`http://geuz.org/getdp`) to see how this is done.

2. Can I launch Gmsh from my solver (instead of launching my solver from Gmsh) in order to monitor a solution?

   Sure. The simplest (but rather crude) approach if to re-launch Gmsh everytime you want to visualize something (a simple C program showing how to do this is given in '`utils/misc/callgmsh.c`'). A better approach is to modify your program so that it can communicate with Gmsh over a socket (see "How do I integrate my own solver with Gmsh?" above; you can skip the option file creation). Then select 'Always listen to incoming connection requests' in the solver option panel (or run gmsh with the `-listen` command line option) and Gmsh will always listen for your program on the Solver.SocketName socket.

D.7 Post-processing module

1. How do I compute a section of a plot?

   Use 'Tools->Plugins->Cut Plane'.

2. Can I save an isosurface to a file?

   Yes: first run 'Tools->Plugins->Cut Map' to extract the isosurface, then use 'View->Save As' to save the new view.

3. Can Gmsh generate isovolumes?

   Yes, with the CutMap plugin (set the ExtractVolume option to -1 or 1 to extract the negative or positive levelset).

4. How do I animate my plots?

   If the views contain multiple time steps, you can press the 'play' button at the bottom of the graphic window, or change the time step by hand in the view option panel. You can also use the left and right arrow keys on your keyboard to change the time step in all visible views in real time.

If you want to loop through different views instead of time steps, you can use the 'Loop through views instead of time steps' option in the view option panel, or use the up and down arrow keys on your keyboard.

5. How do I visualize a deformed mesh?

   Load a vector view containing the displacement field, and set 'Vector display' to 'Displacement' in 'View->Options->Aspect'. If the displacement is too small (or too large), you can scale it with the 'Displacement factor' option. (Remember that you can drag the mouse in all numeric input fields to slide the value!)

   Another option is to use the 'General transformation expressions' (in View->Options->Offset) on a scalar view, with the displacement map selected as the data source.

6. Can I visualize a field on a deformed mesh?

   Yes, there are several ways to do that.

   The easiest is to load two views: the first one containing a displacement field (a vector view that will be used to deform the mesh), and the second one containing the field you want to display (this view has to contain the same number of elements as the displacement view). You should then set 'Vector display' to 'Displacement' in the first view, as well as set 'Data source' to point to the second view. (You might want to make the second view invisible, too. If you want to amplify or decrease the amount of deformation, just modify the 'Displacement factor' option.)

   Another solution is to use the 'General transformation expressions' (in 'View->Options->Offset') on the field you want to display, with the displacement map selected as the data source.

   And yet another solution is to use the Warp plugin.

7. Can I color the arrows representing a vector field with data from a scalar field?

   Yes: load both the vector and the scalar fields (the two views must have the same number of elements) and, in the vector field options, select the scalar view in 'Data source'.

8. Can I color isovalue surfaces with data from another scalar view?

   Yes, using either the CutMap plugin (with the 'dView' option) or the Evaluate plugin.

9. Is there a way to save animations?

   Yes, using scripts. Have a look at '`tutorial/t8.geo`' or '`demos/anim.script`' for some examples.

10. Is there a way to visualize only certain components of vector/tensor fields?

    Yes, by using either the "Force field" options in 'Tools->Options->View->Visibility', or by using 'Tools->Plugins->MathEval'.

11. Can I do arithmetic operations on a view? Can I perform operations involving different views?

    Yes, with the Evaluate plugin.

12. Some plugins seem to create empty views. What's wrong?

    There can be several reasons:

    - the plugin might be written for specific element types only (for example, only for scalar triangles or tetrahedra). In that case, you should transform your view before running the plugin (you can use `Plugin(DecomposeinSimplex)` to transform all quads, hexas, prisms and pyramids into triangles and tetrahedra).

- the plugin might expect a mesh while all you provide is a point cloud. In 2D, you can use `Plugin(Triangulate)` to transform a point cloud into a triangulated surface. In 3D you can use `Plugin(Tetrahedralize)`.
- the input parameters are out of range.

In any case, you can automatically remove all empty views with 'View->Remove->Empty Views' in the GUI, or with `Delete Empty Views;` in a script.

13. How can I see "inside" a complicated post-processing view?

    Use 'Tools->Clipping Planes'.

    When viewing 3D scalar fields, you can also modify the colormap ('Tools->Options->View->Map') to make the iso-surfaces "transparent": either by holding 'Ctrl' while dragging the mouse to draw the alpha channel by hand, or by using the 'a', 'Ctrl+a', 'p' and 'Ctrl+p' keyboard shortcuts.

    Yet another (destructive) option is to use the ExtractVolume option in the CutSphere or CutPlane plugins.

14. I am loading a valid 3D scalar view but Gmsh does not display anything!

    If your dataset is constant per element make sure you don't use the 'Iso-values' interval type in 'Tools->Options->View->Range'.

Appendix E Version history

2.7.0 (March 9, 2013): new single-window GUI, with dynamically customizable widget tree; faster STEP/BRep import; arbitrary size image export; faster 2D Delaunay/Frontal algorithms; full option viewer/editor; many bug fixes.

2.6.1 (July 15, 2012): minor improvements and bug fixes.

2.6.0 (Jun 19, 2012): new quadrilateral meshing algorithms (Blossom and DelQuad); new solver module based on ONELAB project (requires FLTK 1.3); new tensor field visualization modes (eigenvectors, ellipsoid, etc.); added support for interpolation schemes in .msh file; added support for MED3 format; rescale viewport around visible entities (shift+1:1 in GUI); unified post-processing field export; new experimental stereo+camera visualization mode; added experimental BAMG & MMG3D support for anisotropic mesh generation; new OCC cut & merge algorithm imported from Salome; new ability to connect extruded meshes to tetrahedral grids using pyramids; new homology solver; Abaqus (INP) mesh export; new Python and Java wrappers; bug fixes and small improvements all over the place.

2.5.0 (Oct 15, 2010): new compound geometrical entities (for remeshing and/or trans-patch meshing); improved mesh reclassification tool; new client/server visualization mode; new ability to watch a pattern of files to merge; new integrated MPEG export; new option to force the type of views dynamically; bumped mesh version format to 2.2 (small change in the meaning of the partition tags; this only affects partitioned (i.e. parallel) meshes); renamed several post-processing plugins (as well as plugin options) to make them easier to understand; many bug fixes and usability improvements all over the place.

2.4.2 (Sep 21, 2009): solver code refactoring + better IDE integration.

2.4.1 (Sep 1, 2009): fixed surface mesh orientation bug introduced in 2.4.0; mesh and graphics code refactoring, small usability enhancements and bug fixes.

2.4.0 (Aug 22, 2009): switched build system to CMake; optionally copy transfinite mesh contraints during geometry transformations; bumped mesh version format to 2.1 (small change in the $PhysicalNames section, where the group dimension is now required); ported most plugins to the new post-processing API; switched from MathEval to MathEx and Flu_Tree_Browser to Fl_Tree; small bug fixes and improvements all over the place.

2.3.1 (Mar 18, 2009): removed GSL dependency (Gmsh now simply uses Blas and Lapack); new per-window visibility; added support for composite window printing and background images; fixed string option affectation in parser; fixed surface mesh orientation for Open CASCADE models; fixed random triangle orientations in Delaunay and Frontal algorithms.

```
2.3.0 (Jan 23, 2009): major graphics and GUI code refactoring; new
full-quad/hexa subdivision algorithm; improved automatic transfinite corner
selection (now also for volumes); improved visibility browser; new automatic
adaptive visualization for high-order simplices; modified arrow size, clipping
planes and transform options; many improvements and bug fixes all over the
place.

2.2.6 (Nov 21, 2008): better transfinite smoothing and automatic corner
selection; fixed high order meshing crashes on Windows and Linux; new uniform
mesh refinement (thanks Brian!); fixed various other small bugs.

2.2.5 (Oct 25, 2008): Gmsh now requires FLTK 1.1.7 or above; various small
improvements (STL and VTK mesh IO, Netgen upgrade, Visual C++ support, Fields,
Mesh.{Msh,Stl,...}Binary changed to Mesh.Bindary) and bug fixes (pyramid
interpolation, Chaco crashes).

2.2.4 (Aug 14, 2008): integrated Metis and Chaco mesh partitioners; variables
can now be deleted in geo files; added support for point datasets in model-based
postprocessing views; small bug fixes.

2.2.3 (Jul 14, 2008): enhanced clipping interface; API cleanup; fixed various
bugs (Plugin(Integrate), high order meshes, surface info crash).

2.2.2 (Jun 20, 2008): added geometrical transformations on volumes; fixed bug in
high order mesh generation.

2.2.1 (Jun 15, 2008): various small improvements (adaptive views, GUI, code
cleanup) and bug fixes (high order meshes, Netgen interface).

2.2.0 (Apr 19, 2008): new model-based post-processing backend; added MED I/O for
mesh and post-processing; fixed BDF vertex ordering for 2nd order elements;
replaced Mesh.ConstrainedBackgroundMesh with
Mesh.CharacteristicLength{FromPoints,ExtendFromBoundary}; new Fields interface;
control windows are now non-modal by default; new experimental 2D frontal
algorithm; fixed various bugs.

2.1.1 (Mar 1, 2008): small bug fixes (second order meshes, combine views, divide
and conquer crash, ...).

2.1.0 (Feb 23, 2008): new post-processing database; complete rewrite of
post-processing drawing code; improved surface mesh algorithms; improved
STEP/IGES/BREP support; new 3D mesh optimization algorithm; new default native
file choosers; fixed 'could not find extruded vertex' in extrusions; many
improvements and bug fixes all over the place.

2.0.8 (Jul 13, 2007): unused vertices are not saved in mesh files anymore; new
plugin GUI; automatic GUI font size selection; renamed
```

```
Plugin(DecomposeInSimplex) into Plugin(MakeSimplex); reintroduced enhanced
Plugin(SphericalRaise); clarified meshing algo names; new option to save groups
of nodes in UNV meshes; new background mesh infrastructure; many small
improvements and small bug fixes.

2.0.7 (Apr 3, 2007): volumes can now be defined from external CAD surfaces;
Delaunay/Tetgen algorithm is now used by default when available; re-added
support for Plot3D structured mesh format; added ability to export external CAD
models as GEO files (this only works for the limited set of geometrical
primitives available in the GEO language, of course--so trying to convert e.g. a
trimmed NURBS from a STEP file into a GEO file will fail); "lateral" entities
are now added at the end of the list returned by extrusion commands; fixed
various bugs.

2.0 (Feb 5, 2007): new geometry and mesh databases, with support for STEP and
IGES import via Open CASCADE; complete rewrite of geometry and mesh drawing
code; complete rewrite of mesh I/O layer (with new native binary MSH format and
support for import/export of I-deas UNV, Nastran BDF, STL, Medit MESH and VRML
1.0 files); added support for incomplete second order elements; new 2D and 3D
meshing algorithms; improved integration of Netgen and TetGen algorithms;
removed anisotropic meshing algorithm (as well as attractors); removed explicit
region number specification in extrusions; option changes in the graphical
interface are now applied instantaneously; added support for offscreen rendering
using OSMesa; added support for SVG output; added string labels for Physical
entities; lots of other improvements all over the place.

1.65 (May 15, 2006): new Plugin(ExtractEdges); fixed compilation errors with
gcc4.1; replaced Plugin(DisplacementRaise) and Plugin(SphericalRaise) with the
more flexible Plugin(Warp); better handling of discrete curves; new Status
command in parser; added option to renumber nodes in .msh files (to avoid holes
in the numbering sequence); fixed 2 special cases in quad->prism extrusion;
fixed saving of 2nd order hexas with negative volume; small bug fixes and
cleanups.

1.64 (Mar 18, 2006): Windows versions do no depend on Cygwin anymore; various
bug fixes and cleanups.

1.63 (Feb 01, 2006): post-processing views can now be exported as meshes;
improved background mesh handling (a lot faster, and more accurate); improved
support for input images; new Plugin(ExtractElements); small bug fixes and
enhancements.

1.62 (Jan 15, 2006): new option to draw color gradients in the background;
enhanced perspective projection mode; new "lasso" selection mode (same as
"lasso" zoom, but in selection mode); new "invert selection" button in the
visibility browser; new snapping grid when adding points in the GUI; nicer
normal smoothing; new extrude syntax (old syntax still available, but
```

deprecated); various small bug fixes and enhancements.

1.61 (Nov 29, 2005): added support for second order (curved) elements in post-processor; new version (1.4) of post-processing file formats; new stippling options for 2D plots; removed limit on allowed number of files on command line; all "Combine" operations are now available in the parser; changed View.ArrowLocation into View.GlyphLocation; optimized memory usage when loading many (>1000) views; optimized loading and drawing of line meshes and 2D iso views; optimized handling of meshes with large number of physical entities; optimized vertex array creation for large post-processing views on Windows/Cygwin; removed Discrete Line and Discrete Surface commands (the same functionality can now be obtained by simply loading a mesh in .msh format); fixed coloring by mesh partition; added option to light wireframe meshes and views; new "mesh statistics" export format; new full-quad recombine option; new Plugin(ModulusPhase); hexas and prisms are now always saved with positive volume; improved interactive entity selection; new experimental Tetgen integration; new experimental STL remeshing algorithm; various small bug fixes and improvements.

1.60 (Mar 15, 2005): added support for discrete curves; new Window menu on Mac OS X; generalized all octree-based plugins (CutGrid, StreamLines, Probe, etc.) to handle all element types (and not only scalar and vector triangles+tetrahedra); generalized Plugin(Evaluate), Plugin(Extract) and Plugin(Annotate); enhanced clipping plane interface; new grid/axes/rulers for 3D post-processing views (renamed the AbscissaName, NbAbscissa and AbscissaFormat options to more general names in the process); better automatic positioning of 2D graphs; new manipulator dialog to specify rotations, translations and scalings "by hand"; various small enhancements and bug fixes.

1.59 (Feb 06, 2005): added support for discrete (triangulated) surfaces, either in STL format or with the new "Discrete Surface" command; added STL and Text output format for post-processing views and STL output format for surface meshes; all levelset-based plugins can now also compute isovolumes; generalized Plugin(Evaluate) to handle external view data (based on the same or on a different mesh); generalized Plugin(CutGrid); new plugins (Eigenvalues, Gradient, Curl, Divergence); changed default colormap to match Matlab's "Jet" colormap; new transformation matrix option for views (for non-destructive rotations, symmetries, etc.); improved solver interface to keep the GUI responsive during solver calls; new C++ and Python solver examples; simplified Tools->Visibility GUI; transfinite lines with "Progression" now allow negative line numbers to reverse the progression; added ability to retrieve Gmsh's version number in the parser (to help write backward compatible scripts); fixed white space in unv mesh output; fixed various small bugs.

1.58 (Jan 01, 2005): fixed UNIX socket interface on Windows (broken by the TCP solver patch in 1.57); bumped version number of default post-processing file formats to 1.3 (the only small modification is the handling of the end-of-string character for text2d and text3d objects in the ASCII format); new File->Rename menu; new colormaps+improved colormap handling; new color+min/max options in views; new GetValue() function to ask for values interactively in scripts; generalized For/EndFor loops in parser; new plugins (Annotate, Remove, Probe); new text attributes in views; renamed some shortcuts; fixed TeX output for large scenes; new option dialogs for various output formats; fixed many small memory leaks in parser; many small enhancements to polish the graphics and the user interface.

1.57 (Dec 23, 2004): generalized displacement maps to display arbitrary view types; the arrows representing a vector field can now also be colored by the values from other scalar, vector or tensor fields; new adaptive high order visualization mode; new options (Solver.SocketCommand, Solver.NameCommand, View.ArrowSizeProportional, View.Normals, View.Tangents and General.ClipFactor); fixed display of undesired solver plugin popups; enhanced interactive plugin behavior; new plugins (HarmonicToTime, Integrate, Eigenvectors); tetrahedral mesh file reading speedup (50% faster on large meshes); large memory footprint reduction (up to 50%) for the visualization of triangular/tetrahedral meshes; the solver interface now supports TCP/IP connections; new generalized raise mode (allows to use complex expressions to offset post-processing maps); upgraded Netgen kernel to version 4.4; new optional TIME list in parsed views to specify the values of the time steps; several bug fixes in the Elliptic mesh algorithm; various other small bug fixes and enhancements.

1.56 (Oct 17, 2004): new post-processing option to draw a scalar view raised by a displacement view without using Plugin(DisplacementRaise) (makes drawing arbitrary scalar fields on deformed meshes much easier); better post-processing menu (arbitrary number of views+scrollable+show view number); improved view->combine; new horizontal post-processing scales; new option to draw the mesh nodes per element; views can now also be saved in "parsed" format; fixed various path problems on Windows; small bug fixes.

1.55 (Aug 21, 2004): added background mesh support for Triangle; meshes can now be displayed using "smoothed" normals (like post-processing views); added GUI for clipping planes; new interactive clipping/cutting plane definition; reorganized the Options GUI; enhanced 3D iso computation; enhanced lighting; many small bug fixes.

1.54 (Jul 03, 2004): integrated Netgen (3D mesh quality optimization + alternative 3D algorithm); Extrude Surface now always automatically creates a new volume (in the same way Extrude Point or Extrude Line create new lines and surfaces, respectively); fixed UNV output; made the "Layers" region numbering consistent between lines, surfaces and volumes; fixed home directory problem on Win98; new Plugin(CutParametric); the default project file is now created in the home directory if no current directory is defined (e.g., when double-clicking on the icon on Windows/Mac); fixed the discrepancy between the orientation of geometrical surfaces and the associated surface meshes; added automatic

```
orientation of surfaces in surface loops; generalized Plugin(Triangulate) to
handle vector and tensor views; much nicer display of discrete iso-surfaces and
custom ranges using smooth normals; small bug fixes and cleanups.

1.53 (Jun 04, 2004): completed support for second order elements in the mesh
module (line, triangles, quadrangles, tetrahedra, hexahedra, prisms and
pyramids); various background mesh fixes and enhancements; major performance
improvements in mesh and post-processing drawing routines (OpenGL vertex arrays
for tri/quads); new Plugin(Evaluate) to evaluate arbitrary expressions on
post-processing views; generalized Plugin(Extract) to handle any combination of
components; generalized "Coherence" to handle transfinite surface/volume
attributes; plugin options can now be set in the option file (like all other
options); added "undo" capability during geometry creation; rewrote the contour
guessing routines so that entities can be selected in an arbitrary order; Mac
users can now double click on geo/msh/pos files in the Finder to launch Gmsh;
removed support for FLTK 1.0; rewrote most of the code related to quadrangles;
fixed 2d elliptic algorithm; removed all OpenGL display list code and options;
fixed light positioning; new BoundingBox command to set the bounding box
explicitly; added support for inexpensive "fake" transparency mode; many code
cleanups.

1.52 (May 06, 2004): new raster ("bitmap") PostScript/EPS/PDF output formats;
new Plugin(Extract) to extract a given component from a post-processing view;
new Plugin(CutGrid) and Plugin(StreamLines); improved mesh projection on
non-planar surfaces; added support for second order tetrahedral elements; added
interactive control of element order; refined mesh entity drawing selection (and
renamed most of the corresponding options); enhanced log scale in
post-processing; better font selection; simplified View.Raise{X,Y,Z} by removing
the scaling; various bug fixes (default postscript printing mode, drawing of 3D
arrows/cylinders on Linux, default home directory on Windows, default initial
file browser directory, extrusion of points with non-normalized axes of
rotation, computation of the scene bounding box in scripts, + the usual
documentation updates).

1.51 (Feb 29, 2004): initial support for visualizing mesh partitions; integrated
version 2.0 of the MSH mesh file format; new option to compute post-processing
ranges (min/max) per time step; Multiple views can now be combined into multi
time step ones (e.g. for programs that generate data one time step at a time);
new syntax: #var[] returns the size of the list var[]; enhanced "gmsh -convert";
temporary and error files are now created in the home directory to avoid file
permission issues; new 3D arrows; better lighting support; STL facets can now be
converted into individual geometrical surfaces; many other small improvements
and bug fixes (multi timestep tensors, color by physical entity, parser cleanup,
etc.).

1.50 (Dec 06, 2003): small changes to the visibility browser + made visibility
scriptable (new Show/Hide commands); fixed (rare) crash when deleting views;
```

split File->Open into File->Open and File->New to behave like most other programs; Mac versions now use the system menu bar by default (if possible); fixed bug leading to degenerate and/or duplicate tetrahedra in extruded meshes; fixed crash when reloading sms meshes.

1.49 (Nov 30, 2003): made Merge, Save and Print behave like Include (i.e., open files in the same directory as the main project file if the path is relative); new Plugin(DecomposeInSimplex); new option View.AlphaChannel to set the transparency factor globally for a post-processing view; new "Combine Views" command; various bug fixes and cleanups.

1.48 (Nov 23, 2003): new DisplacementRaise plugin to plot arbitrary fields on deformed meshes; generalized CutMap, CutPlane, CutSphere and Skin plugins to handle all kinds of elements and fields; new "Save View[n]" command to save views from a script; many small bug fixes (configure tests for libpng, handling of erroneous options, multi time step scalar prism drawings, copy of surface mesh attributes, etc.).

1.47 (Nov 12, 2003): fixed extrusion of surfaces defined by only two curves; new syntax to retrieve point coordinates and indices of entities created through geometrical transformations; new PDF and compressed PostScript output formats; fixed numbering of elements created with "Extrude Point/Line"; use $GMSH_HOME as home directory if defined.

1.46 (Aug 23, 2003): fixed crash for very long command lines; new options for setting the displacement factor and Triangle's parameters + renamed a couple of options to more sensible names (View.VectorType, View.ArrowSize); various small bug fixes; documentation update.

1.45 (Jun 14, 2003): small bug fixes (min/max computation for tensor views, missing physical points in read mesh, "jumping" geometry during interactive manipulation of large models, etc.); variable definition speedup; restored support for second order elements in one- and two-dimensional meshes; documentation updates.

1.44 (Apr 21, 2003): new reference manual; added support for PNG output; fixed small configure script bugs.

1.43 (Mar 28, 2003): fixed solver interface problem on Mac OS X; new option to specify the interactive rotation center (default is now the pseudo "center of gravity" of the object, instead of (0,0,0)).

1.42 (Mar 19, 2003): suppressed the automatic addition of a ".geo" extension if the file given on the command line is not recognized; added missing Layer option for Extrude Point; fixed various small bugs.

1.41 (Mar 04, 2003): Gmsh is now licensed under the GNU General Public License;

```
general code cleanup (indent).

1.40 (Feb 26, 2003): various small bug fixes (mainly GSL-related).

1.39 (Feb 23, 2003): removed all non-free routines; more build system work;
implemented Von-Mises tensor display for all element types; fixed small GUI
bugs.

1.38 (Feb 17, 2003): fixed custom range selection for 3D iso graphs; new build
system based on autoconf; new image reading code to import bitmaps as
post-processing views.

1.37 (Jan 25, 2003): generalized smoothing and cuts of post-processing views;
better Windows integration (solvers, external editors, etc.); small bug fixes.

1.36 (Nov 20, 2002): enhanced view duplication (one can now use "Duplicata
View[num]" in the input file); merged all option dialogs in a new general option
window; enhanced discoverability of the view option menus; new 3D point and line
display; many small bug fixes and enhancements ("Print" format in parser,
post-processing statistics, smooth normals, save window positions, restore
default options, etc.).

1.35 (Sep 11, 2002): graphical user interface upgraded to FLTK 1.1 (tooltips,
new file chooser with multiple selection, full keyboard navigation, cut/paste of
messages, etc.); colors can be now be directly assigned to mesh entities;
initial tensor visualization; new keyboard animation (right/left arrow for time
steps; up/down arrow for view cycling); new VRML output format for surface
meshes; new plugin for spherical elevation plots; new post-processing file
format (version 1.2) supporting quadrangles, hexahedra, prisms and pyramids;
transparency is now enabled by default for post-processing plots; many small bug
fixes (read mesh, ...).

1.34 (Feb 18, 2002): improved surface mesh of non-plane surfaces; fixed
orientation of elements in 2D anisotropic algorithm; minor user interface polish
and additions (mostly in post-processing options); various small bug fixes.

1.33 (Jan 24, 2002): new parameterizable solver interface (allowing up to 5
user-defined solvers); enhanced 2D aniso algorithm; 3D initial mesh speedup.

1.32 (Oct 04, 2001): new visibility browser; better floating point exception
checks; fixed infinite looping when merging meshes in project files; various
small clean ups (degenerate 2D extrusion, view->reload, ...).

1.31 (Nov 30, 2001): corrected ellipses; PostScript output update (better
shading, new combined PS/LaTeX output format); more interface polish; fixed
extra memory allocation in 2D meshes; Physical Volume handling in unv format;
various small fixes.
```

```
1.30 (Nov 16, 2001): interface polish; fix crash when extruding quadrangles.

1.29 (Nov 12, 2001): translations and rotations can now be combined in
extrusions; fixed coherence bug in Extrude Line; various small bug fixes and
additions.

1.28 (Oct 30, 2001): corrected the 'Using Progression' attribute for tranfinite
meshes to actually match a real geometric progression; new Triangulate plugin;
new 2D graphs (space+time charts); better performance of geometrical
transformations (warning: the numbering of some automatically created entities
has changed); new text primitives in post-processing views (file format updated
to version 1.1); more robust mean plane computation and error checks; various
other small additions and clean-ups.

1.27 (Oct 05, 2001): added ability to extrude curves with Layers/Recombine
attributes; new PointSize/LineWidth options; fixed For/EndFor loops in included
files; fixed error messages (line numbers+file names) in loops and functions;
made the automatic removal of duplicate geometrical entities optional
(Geometry.AutoCoherence=0); various other small bug fixes and clean-ups.

1.26 (Sep 06, 2001): enhanced 2D anisotropic mesh generator (metric
intersections); fixed small bug in 3D initial mesh; added alternative syntax for
built-in functions (for GetDP compatibility); added line element display; Gmsh
now saves all the elements in the mesh if no physical groups are defined (or if
Mesh.SaveAll=1).

1.25 (Sep 01, 2001): fixed bug with mixed recombined/non-recombined extruded
meshes; Linux versions are now build with no optimization, due to bugs in gcc
2.95.X.

1.24 (Aug 30, 2001): fixed characteristic length interpolation for Splines;
fixed edge swapping bug in 3D initial mesh; fixed degenerated case in
geometrical extrusion (ruled surface with 3 borders); fixed generation of
degenerated hexahedra and prisms for recombined+extruded meshes; added BSplines
creation in the GUI; integrated Jonathan Shewchuk's Triangle as an alternative
isotropic 2D mesh generator; added AngleSmoothNormals to control sharp edge
display with smoothed normals; fixed random crash for lighted 3D iso surfaces.

1.23 (Aug, 2001): fixed duplicate elements generation + non-matching tetrahedra
faces in 3D extruded meshes; better display of displacement maps; fixed
interactive ellipsis construction; generalized boundary operator; added new
explode option for post-processing views; enhanced link view behavior (to update
only the changed items); added new default plugins: Skin, Transform, Smooth;
fixed various other small bugs (mostly in the post-processing module and for
extruded meshes).
```

```
1.22 (Aug 03, 2001): fixed (yet another) bug for 2D mesh in the mean plane;
fixed surface coherence bug in extruded meshes; new double logarithmic scale,
saturate value and smoothed normals option for post-processing views; plugins
are now enabled by default; three new experimental statically linked plugins:
CutMap (extracts a given iso surface from a 3D scalar map), CutPlane (cuts a 3D
scalar map with a plane section), CutSphere (cuts a 3D scalar map with a
sphere); various other bug fixes, additions and clean-ups.

1.21 (Jul 25, 2001): fixed more memory leaks; added -opt command line option to
parse definitions directly from the command line; fixed missing screen refreshes
during contour/surface/volume selection; enhanced string manipulation functions
(Sprintf, StrCat, StrPrefix); many other small fixes and clean-ups.

1.20 (Jun 14, 2001): fixed various bugs (memory leaks, functions in included
files, solver command selection, ColorTable option, duplicate nodes in extruded
meshes (not finished yet), infinite loop on empty views, orientation of
recombined quadrangles, ...); reorganized the interface menus; added constrained
background mesh and mesh visibility options; added mesh quality histograms;
changed default mesh colors; reintegrated the old command-line extrusion mesh
generator.

1.19 (May 07, 2001): fixed seg. fault for scalar simplex post-processing; new
Solver menu; interface for GetDP solver through sockets; fixed multiple scale
alignment; added some options + full option descriptions.

1.18 (Apr 26, 2001): fixed many small bugs and incoherences in post-processing;
fixed broken background mesh in 1D mesh generation.

1.17 (Apr 17, 2001): corrected physical points saving; fixed parsing of DOS
files (carriage return problems); easier geometrical selections (cursor change);
plugin manager; enhanced variable arrays (sublist selection and affectation);
line loop check; New arrow display; reduced number of 'fatal' errors + better
handling in interactive mode; fixed bug when opening meshes; enhanced File->Open
behavior for meshes and post-processing views.

1.16 (Feb 26, 2001): added single/double buffer selection (only useful for Unix
versions of Gmsh run from remote hosts without GLX); fixed a bug for recent
versions of the opengl32.dll on Windows, which caused OpenGL fonts not to show
up.

1.15 (Feb 23, 2001): added automatic visibility setting during entity selection;
corrected geometrical extrusion bug.

1.14 (Feb 17, 2001): corrected a few bugs in the GUI (most of them were
introduced in 1.13); added interactive color selection; made the option database
bidirectional (i.e. scripts now correctly update the GUI); default options can
now be saved and automatically reloaded at startup; made some changes to the
```

scripting syntax (PostProcessing.View[n] becomes View[n]; Offset0 becomes OffsetX, etc.); corrected the handling of simple triangular surfaces with large characteristic lengths in the 2D isotropic algorithm; added an ASCII to binary post-processing view converter.

1.13 (Feb 09, 2001): added support for JPEG output on Windows.

1.12: corrected vector lines in the post-processing parsed format; corrected animation on Windows; corrected file creation in scripts on Windows; direct affectation of variable arrays.

1.11 (Feb 07, 2001): corrected included file loading problem.

1.10 (Feb 04, 2001): switched from Motif to FLTK for the GUI. Many small tweaks.

1.00 (Jan 15, 2001): added PPM and YUV output; corrected nested If/Endif; Corrected several bugs for pixel output and enhanced GIF output (dithering, transparency); slightly changed the post-processing file format to allow both single and double precision numbers.

0.999 (Dec 20, 2000): added JPEG output and easy MPEG generation (see t8.geo in the tutorial); clean up of export functions; small fixes; Linux versions are now compiled with gcc 2.95.2, which should fix the problems encountered with Mandrake 7.2.

0.998 (Dec 19, 2000): corrected bug introduced in 0.997 in the generation of the initial 3D mesh.

0.997 (Dec 14, 2000): corrected bug in interactive surface/volume selection; Added interactive symmetry; corrected geometrical extrusion with rotation in degenerated or partially degenerated cases; corrected bug in 2D mesh when meshing in the mean plane.

0.996: arrays of variables; enhanced Printf and Sprintf; Simplified options (suppression of option arrays).

0.995 (Dec 11, 2000): totally rewritten geometrical database (performance has been drastically improved for all geometrical transformations, and most notably for extrusion). As a consequence, the internal numbering of geometrical entities has changed: this will cause incompatibilities with old .geo files, and will require a partial rewrite of your old .geo files if these files made use of geometrical transformations. The syntax of the .geo file has also been clarified. Many additions for scripting purposes. New extrusion mesh generator. Preliminary version of the coupling between extruded and Delaunay meshes. New option and procedural database. All interactive operations can be scripted in the input files. See the last example in the tutorial for an example. Many stability enhancements in the 2D and 3D mesh algorithms. Performance boost of the 3D algorithm. Gmsh is still slow, but the performance becomes acceptable. An average 1000 tetrahedra/second is obtained on a 600Mhz computer for a mesh of one million tetrahedra. New anisotropic 2D mesh algorithm. New (ASCII and binary) post-processing file format and clarified mesh file format. New handling for interactive rotations (trackball mode). New didactic interactive mesh construction (watch the Delaunay algorithm in real time on complex geometries: that's exciting ;-). And many, many bug fixes and cleanups.

0.992 (Nov 13, 2000): corrected recombined extrusion; corrected ellipses; added simple automatic animation of post-processing maps; fixed various bugs.

0.991 (Oct 24, 2000): fixed a serious allocation bug in 2D algorithm, which caused random crashes. All users should upgrade to 0.991.

0.990: bug fix in non-recombined 3D transfinite meshes.

0.989 (Sep 01, 2000): added ability to reload previously saved meshes; some new command line options; reorganization of the scale menu; GIF output.

0.987: fixed bug with smoothing (leading to the possible generation of erroneous 3d meshes); corrected bug for mixed 3D meshes; moved the 'toggle view link' option to Opt->Postprocessing_Options.

0.986: fixed overlay problems; SGI version should now also run on 32 bits machines; fixed small 3d mesh bug.

0.985: corrected colormap bug on HP, SUN, SGI and IBM versions; corrected small initialization bug in postscript output.

0.984: corrected bug in display lists; added some options in Opt->General.

0.983: corrected some seg. faults in interactive mode; corrected bug in rotations; changed default window sizes for better match with 1024x768 screens (default X resources can be changed: see ex03.geo).

0.982: lighting for mesh and post-processing; corrected 2nd order mesh on non plane surfaces; added example 13.

Appendix F Copyright and credits

```
                 Gmsh is copyright (C) 1997-2013

                       Christophe Geuzaine
                     <cgeuzaine at ulg.ac.be>

                               and

                      Jean-Francois Remacle
              <jean-francois.remacle at uclouvain.be>
```

Code contributions to Gmsh have been provided by David Colignon (colormaps), Emilie Marchandise (compound geometrical entities), Gaetan Bricteux (Gauss integration and levelsets), Jacques Lechelle (DIFFPACK mesh format), Jonathan Lambrechts (fields, solver), Jozef Vesely (help with Tetgen integration), Koen Hillewaert (high order elements), Laurent Stainier (eigenvalue solvers, tensor display and MacOS bits), Marc Ume (original list code), Mark van Doesburg (Open CASCADE face connection), Matt Gundry (Plot3d mesh format), Matti Pellikka (Cell complex and Homology solver), Nicolas Tardieu (help with Netgen integration), Pascale Noyret (MED mesh format), Pierre Badel (root finding and minimization), Ruth Sabariego (pyramids), Stephen Guzik (CGNS and partitioners), Bastien Gorissen (parallel remote post-processing), Eric Bechet (solver), Gilles Marckmann (camera and stero mode), Ashish Negi (Salome/Netgen CAD healing), Trevor Strickler (structured/unstructured coupling with pyramids), Amaury Johnen (Bezier), Benjamin Ruard (Java wrappers). See comments in the sources for more information. If we forgot to list your contributions please send us an email!

Special thanks to Bill Spitzak, Michael Sweet, Matthias Melcher, Greg Ercolano
and others for the Fast Light Tool Kit on which Gmsh's GUI is based. See
http://www.fltk.org for more info on this excellent object-oriented,
cross-platform toolkit.

Special thanks also to EDF for funding the Open CASCADE and MED integration.

Thanks to the following folks who have contributed by providing fresh ideas on
theoretical or programming topics, who have sent patches, requests for changes
or improvements, or who gave us access to exotic machines for testing Gmsh: Juan
Abanto, Olivier Adam, Guillaume Alleon, Laurent Champaney, Pascal Dupuis,
Patrick Dular, Philippe Geuzaine, Johan Gyselinck, Francois Henrotte, Benoit
Meys, Nicolas Moes, Osamu Nakamura, Chad Schmutzer, Jean-Luc Fl'ejou, Xavier
Dardenne, Christophe Prud'homme, Sebastien Clerc, Jose Miguel Pasini, Philippe
Lussou, Jacques Kools, Bayram Yenikaya, Peter Hornby, Krishna Mohan Gundu,
Christopher Stott, Timmy Schumacher, Carl Osterwisch, Bruno Frackowiak, Philip
Kelleners, Romuald Conty, Renaud Sizaire, Michel Benhamou, Tom De Vuyst, Kris
Van den Abeele, Simon Vun, Simon Corbin, Thomas De-Soza, Marcus Drosson, Antoine
Dechaume, Jose Paulo Moitinho de Almeida, Thomas Pinchard, Corrado Chisari, Axel
Hackbarth, Peter Wainwright, Jiri Hnidek, Thierry Thomas, Konstantinos Poulios,
Laurent Van Miegroet, Shahrokh Ghavamian, Geordie McBain.

Appendix G License

themselves, then this License, and its terms, do not apply to those
sections when you distribute them as separate works. But when you
distribute the same sections as part of a whole which is a work based
on the Program, the distribution of the whole must be on the terms of
this License, whose permissions for other licensees extend to the
entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest
your rights to work written entirely by you; rather, the intent is to
exercise the right to control the distribution of derivative or
collective works based on the Program.

In addition, mere aggregation of another work not based on the Program
with the Program (or with a work based on the Program) on a volume of
a storage or distribution medium does not bring the other work under
the scope of this License.

3. You may copy and distribute the Program (or a work based on it,
under Section 2) in object code or executable form under the terms of
Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable
source code, which must be distributed under the terms of Sections
1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three
years, to give any third party, for a charge no more than your
cost of physically performing source distribution, a complete
machine-readable copy of the corresponding source code, to be
distributed under the terms of Sections 1 and 2 above on a medium
customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer
to distribute corresponding source code. (This alternative is
allowed only for noncommercial distribution and only if you
received the program in object code or executable form with such
an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for
making modifications to it. For an executable work, complete source
code means all the source code for all modules it contains, plus any
associated interface definition files, plus the scripts used to
control compilation and installation of the executable. However, as a
special exception, the source code distributed need not include
anything that is normally distributed (in either source or binary
form) with the major components (compiler, kernel, and so on) of the
operating system on which the executable runs, unless that component

itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), 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 distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under

Concept index

Syntax index