Gmsh

Gmsh Reference Manual

The documentation for Gmsh 2.2
A finite element mesh generator with built-in pre- and post-processing facilities

19 April 2008

Christophe Geuzaine
Jean-François Remacle

Short Contents

Table of Contents

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 C [License], page 193). 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 Section 11.3 [Credits], page 176.

The source code and various pre-compiled versions of Gmsh (for Unix, Windows and Mac OS) can be downloaded from the webpage `http://geuz.org/gmsh/`.

If you use Gmsh, we would appreciate that you mention it in your work. References, as well as the latest news about Gmsh development, are always available on `http://geuz.org/gmsh/`. Please send all Gmsh-related questions to the public Gmsh mailing list at `gmsh@geuz.org`.

If you want to integrate Gmsh into a closed-source software, or want to sell a modified closed-source version of Gmsh, please contact one of the authors. You can purchase a version of Gmsh under a different license, with “no strings attached” (for example allowing you to take parts of Gmsh and integrate them into your own proprietary code).

1 Overview

Gmsh is an automatic three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. Its design goal is to provide a simple meshing tool for academic problems 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 ASCII data 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 ("b-rep") 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. Compound 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 parameterized.

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[1]. This automatically assures the conformity of the mesh when, for example, two surfaces share a common line. But this also implies that

---

[1] 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...

the discretization of an "isolated" (n-1)-th dimensional entity inside an n-th dimensional entity does *not* constrain the n-th dimensional mesh. Every meshing step is constrained by the characteristic length field, which can be uniform, specified by characteristic lengths associated with points in the geometry, or defined by general "fields" (a scalar field defined on another mesh using post-processing view, threshold fields associated with point or line "attractors", etc.).

For each meshing step, all structured mesh directives are executed first, and serve as additional constraints for the unstructured parts [2].

1.3 Solver: external solver interface

External solvers can be interfaced with Gmsh through Unix or TCP/IP sockets, which permits to launch external computations and to collect and process the results directly from within Gmsh's post-processing module. The default solver interfaced with Gmsh is GetDP (`http://www.geuz.org/getdp/`). Examples on how to interface solvers written in C, C++, Perl and Python 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 ASCII text 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 . . .

Gmsh is a (relatively) small program, and was principally developed "in academia, to solve academic problems". . . Nevertheless, over the years, many people outside universities have found Gmsh useful in their day-to-day jobs. 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 2.4 [User-defined functions], page 12, Section 2.5 [Loops and conditionals], page 12, and Section 2.6 [General commands], page 13);

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

- parameterize these geometries. Gmsh's scripting language enables all commands and command arguments to depend on previous calculations (see Section 2.1 [Expressions], page 7, and Section 3.1 [Geometry commands], page 41);
- import complex models in industry-standard formats like STEP or IGES (when Gmsh is built with OpenCascade support);
- generate 1D, 2D and 3D simplicial (i.e., using line segments, triangles and tetrahedra) finite element meshes. The performance of the 1D and 2D algorithms is pretty good; the 3D algorithm is still experimental and slow (see Chapter 4 [Mesh module], page 53, and Chapter 7 [Tutorial], page 121);
- 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 geometry points or using flexible mesh size fields (see Section 4.2 [Mesh commands], page 54);
- create simple extruded geometries and meshes (see Section 3.1 [Geometry commands], page 41, and Section 4.2 [Mesh commands], page 54);
- interact with external solvers. Gmsh provides C, C++, Perl and Python interfaces, and others can be easily added (see Chapter 5 [Solver module], page 69);
- visualize computational results in a great variety of ways. Gmsh can display scalar, vector and tensor data sets, and can perform various operations on the resulting post-processing views (see Chapter 6 [Post-processing module], page 87);
- export plots in many different formats: vector PostScript or encapsulated PostScript, LaTeX, PNG, JPEG, ... (see Section 2.7 [General options], page 16);
- generate complex animations (see Chapter 2 [General tools], page 7, and Section 7.8 [t8.geo], page 134);
- 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 8 [Running Gmsh], page 139);
- 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 Section 2.7 [General options], page 16, Section 3.2 [Geometry options], page 46, Section 4.3 [Mesh options], page 57, Section 6.3 [Post-processing options], page 104, and Chapter 8 [Running Gmsh], page 139);
- and do all the above on various platforms (Windows, Mac and Unix), for free (see [Copying conditions], page 1), using clear-text ASCII files and/or a small but powerful graphical user interface.

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

Due to its historical background and limited developer manpower, Gmsh has also some (a lot of?) weaknesses:

- the b-rep approach for describing geometries can become inconvenient for complex models;

- there is no support for Nurbs and only very limited support for trimmed surfaces in Gmsh's scripting language (however you can import STEP or IGES models with such features when Gmsh is built with OpenCascade support);
- Gmsh is not primarily a structured mesh generator: no automatic quadrilateral or hexahedral meshing algorithm is provided. If you want quadrangles, you have to use transfinite or extruded meshes or recombine unstructured triangular meshes. For hexahedra, your only choice is transfinite or extruded meshes;
- Gmsh is not a multi-bloc generator: all meshes produced by Gmsh are conforming in the sense of finite element meshes;
- Gmsh was designed to solve academic "test cases", not industrial-size problems. You may find that Gmsh is too slow for large problems (with thousands of geometric primitives, or millions of mesh/post-processing elements).

If you have the skills and some free time, feel free to join the project! We gladly accept any code contributions (see Chapter 10 [Programming notes], page 163) to remedy the aforementioned (and all other) shortcomings...

1.7 Syntactic rules used in this document

Here are the rules we tried to follow when writing this user's guide. 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.

1.8 Comments

All Gmsh ASCII text input 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.

2 General tools

This chapter describes the general commands and options that can be used in Gmsh's ASCII text input 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 3 [Geometry module], page 41, Chapter 4 [Mesh module], page 53, Chapter 5 [Solver module], page 69, and Chapter 6 [Post-processing module], page 87, respectively.

Note that, if you are just beginning to use Gmsh, or just want to see what Gmsh is all about, you really don't need to read this chapter and the four next ones. Just have a quick look at Chapter 8 [Running Gmsh], page 139, and go play with the graphical user interface, running the tutorials and demonstration files bundled in the distribution! Most of the commands and options described in the following chapters are available interactively in the GUI, so you don't need to worry about Gmsh's internals for creating your first geometries, meshes and post-processing plots. Once you master the tutorial (read the source files: they are heavily commented—see Chapter 7 [Tutorial], page 121), you might want to come back here to learn more about the specific syntax of Gmsh's commands and esoteric options.

2.1 Expressions

The two constant types used in Gmsh 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.

2.1.1 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 1.7 [Syntactic rules], page 6), and are evaluated during the parsing of the data 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 expression |
  built-in-function |
  real-option |
  GetValue("string", expression)
```

Such *expressions* are used in most of Gmsh's 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 2.2 [Operators], page 9. For the definition of *built-in-functions*, see Section 2.3 [Built-in functions], page 11. The various *real-options* are listed in Section 2.7 [General options], page 16, Section 3.2 [Geometry options], page 46, Section 4.3 [Mesh options], page 57, Section 5.1 [Solver options], page 69, and Section 6.3 [Post-processing options], page 104.

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 2.7 [General options], page 16), 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 [ { expression-list } ] |
  Point { expression } |
  transform |
  extrude
```

The second case in this last definition permits to create a list containing the range of numbers comprised between two *expressions*, with a unit incrementation step. The third case also permits to create a list containing the range of numbers comprised between two *expressions*, but with a positive or negative incrementation step equal to the third *expression*. The fourth case permits to reference an expression list. The fifth case permits to reference an expression sublist (whose elements are those corresponding to the indices provided by the *expression-list*). The sixth case permits to retrieve the coordinates of a given geometry point (see Section 3.1.1 [Points], page 41). The last two cases permit to retrieve the indices of entities created through geometrical transformations and extrusions (see Section 3.1.6 [Transformations], page 45, and Section 3.1.5 [Extrusions], page 44).

To see the practical use of such expressions, have a look at the first couple of examples in Chapter 7 [Tutorial], page 121. Note that, in order to lighten the syntax, you can always 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-items*.

2.1.2 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 )
```

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 last case permits to use the value of a *char-option* as a *char-expression*. The various *char-option*s are listed in Section 2.7 [General options], page 16, Section 3.2 [Geometry options], page 46, Section 4.3 [Mesh options], page 57, Section 5.1 [Solver options], page 69, and Section 6.3 [Post-processing options], page 104.

Character expressions are mostly used to specify non-numeric options and input/output file names. See Section 7.8 [t8.geo], page 134, for an interesting usage of *char-expression*s in an animation script.

2.1.3 Color expressions

Colors expressions are hybrids between fixed-length braced *expression-list*s and *string*s:

```
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 Gmsh's source tree 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-option*s are listed in Section 2.7 [General options], page 16, Section 3.2 [Geometry options], page 46, Section 4.3 [Mesh options], page 57, Section 5.1 [Solver options], page 69, and Section 6.3 [Post-processing options], page 104.

See Section 7.3 [t3.geo], page 125, for an example of the use of color expressions.

2.2 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)
4. `*`, `/`, `%`
5. `+`, `-`
6. `<`, `>`, `<=`, `>=`

---

[1] The affectation operators are introduced in Section 2.6 [General commands], page 13.

7. `==`, `!=`
8. `&&`
9. `||`
10. `?:`
11. `=`, `+=`, `-=`, `*=`, `/=`

2.3 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.

`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.

---

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

`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*.

2.4 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 7.5 [t5.geo], page 129, for an example of a user-defined function.

2.5 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 7.5 [t5.geo], page 129, for an example of `For` and `If` commands. Gmsh does not provide any `Else` (or similar) command at the time of this writing.

2.6 General commands

The following commands can be used anywhere in a Gmsh ASCII text input file:

string `=` *expression*`;`
: Creates a new expression identifier *string*, or affects *expression* to an existing expression identifier. Eleven 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 3 [Geometry module], page 41, 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 2.4 [User-defined functions], page 12) 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` 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. (Remember the remark we made when we defined *expression-list*s: the braces enclosing an *expression-list* are optional if the list only contains a single item.)

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-list*s must contain the same number of items.

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.

string `|` *real-option* `+=` *expression*`;`
Adds and affects *expression* to an existing expression identifier or to a real option.

[3] For compatibility purposes, the behavior of `newl`, `news`, `newv` and `newreg` can be modified with the `Geometry.OldNewReg` option (see Section 3.2 [Geometry options], page 46).

string `|` *real-option* `-=` *expression*`;`
: Subtracts and affects *expression* to an existing expression identifier or to a real option.

string `|` *real-option* `*=` *expression*`;`
: Multiplies and affects *expression* to an existing expression identifier or to a real option.

string `|` *real-option* `/=` *expression*`;`
: Divides and affects *expression* to an existing expression identifier or to a real option.

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

`Exit;`
: Aborts the current script.

`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 2.1 [Expressions], page 7), so that only valid floating point formatting characters make sense in *char-expression*. See Section 7.5 [t5.geo], page 129, 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.

`Merge` *char-expression*`;`
: Merges a file named *char-expression*. This command is equivalent to the 'File->Merge' menu in the graphical user interface. 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 All;`
Deletes all geometrical entities and all currently loaded meshes.

`Delete Physicals;`
Deletes all physical groups.

`Print` *char-expression*`;`
Prints the graphic window in a file named *char-expression*, using the current `Print.Format` (see Section 2.7 [General options], page 16). 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.

`Status` *expression*`;`
Forces mesh status to *expression*.

`System` *char-expression*`;`
Executes a system call.

`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.

2.7 General options

Here is the list of the general *char-option*s, *real-option*s and *color-option*s (in that order—check the default values to see the actual types). Most of these options are accessible in the graphical user interface, but not all of them. When running Gmsh interactively, changing an option in the ASCII text input 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.

Gmsh's default behavior is to save some of these options in a per-user "session resource" file (`General.SessionFileName`) 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. Other options can be saved in a per-user "option" file (`General.OptionsFileName`), automatically loaded by Gmsh every time it starts up, by using the 'Tools->Options->Save as defaults' menu.

`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.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.OptionsFileName`
Option file created with 'Tools->Options->Save'; automatically read on startup
Default value: `".gmsh-options"`
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.WebBrowser`
System command to launch a web browser
Default value: `"open %s"`
Saved in: `General.OptionsFileName`

`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.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.BackgroundGradient`
Draw background gradient (0=none, 1=vertical, 2=horizontal, 3=radial)
Default value: `1`
Saved in: `General.OptionsFileName`

`General.Clip0`
Enable clipping plane 0 (Geometry=2^0, Mesh=2^1, View[i]=2^(2+i))
Default value: `0`
Saved in: `-`

`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.Clip1`
: Enable clipping plane 1 (Geometry=2^0, Mesh=2^1, View[i]=2^(2+i))
Default value: `0`
Saved in: `-`

`General.Clip1A`
: First coefficient in equation for clipping plane 1
Default value: `1`
Saved in: `-`

`General.Clip1B`
: Second coefficient in equation for clipping plane 1
Default value: `0`
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.Clip2`
: Enable clipping plane 2 (Geometry=2^0, Mesh=2^1, View[i]=2^(2+i))
Default value: `0`
Saved in: `-`

`General.Clip2A`
: First coefficient in equation for clipping plane 2
Default value: `1`
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: `0`
Saved in: `-`

`General.Clip2D`
Fourth coefficient in equation for clipping plane 2
Default value: `0`
Saved in: `-`

`General.Clip3`
Enable clipping plane 3 (Geometry=2^0, Mesh=2^1, View[i]=2^(2+i))
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: `0`
Saved in: `-`

`General.Clip4`
Enable clipping plane 4 (Geometry=2^0, Mesh=2^1, View[i]=2^(2+i))
Default value: `0`
Saved in: `-`

`General.Clip4A`
First coefficient in equation for clipping plane 4
Default value: `1`
Saved in: `-`

`General.Clip4B`
Second coefficient in equation for clipping plane 4
Default value: `0`
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: `0`
Saved in: `-`

`General.Clip5`
Enable clipping plane 5 (Geometry=2^0, Mesh=2^1, View[i]=2^(2+i))
Default value: `0`
Saved in: `-`

`General.Clip5A`
First coefficient in equation for clipping plane 5
Default value: `1`
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: `0`
Saved in: `-`

`General.Clip5D`
Fourth coefficient in equation for clipping plane 5
Default value: `0`
Saved in: `-`

`General.ClipFactor`
Near and far clipping plane distance factor (decrease value for better z-buffer resolution)
Default value: `5`
Saved in: `-`

`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.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.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.FastRedraw`
Draw simplified model while rotating, panning and zooming
Default value: `0`
Saved in: `General.OptionsFileName`

`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: `17`
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: `600`
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: `1`
Saved in: `-`

`General.MaxY`
Maximum model coordinate along the Y-axis (read-only)
Default value: `1`
Saved in: `-`

`General.MaxZ`
Maximum model coordinate along the Z-axis (read-only)
Default value: `1`
Saved in: `-`

`General.MenuPositionX`
Horizontal position (in pixels) of the upper left corner of the menu window
Default value: `800`
Saved in: `General.SessionFileName`

`General.MenuPositionY`
Vertical position (in pixels) of the upper left corner of the menu window
Default value: `50`
Saved in: `General.SessionFileName`

`General.MessagePositionX`
Horizontal position (in pixels) of the upper left corner of the message window
Default value: `650`
Saved in: `General.SessionFileName`

`General.MessagePositionY`
Vertical position (in pixels) of the upper left corner of the message window
Default value: `490`
Saved in: `General.SessionFileName`

`General.MessageHeight`
Height (in pixels) of the message window
Default value: `180`
Saved in: `General.SessionFileName`

`General.MessageWidth`
Width (in pixels) of the message window
Default value: `100`
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.PluginWidth`
Width (in pixels) of the plugin window
Default value: `100`
Saved in: `General.SessionFileName`

`General.PluginHeight`
Height (in pixels) of the plugin window
Default value: `100`
Saved in: `General.SessionFileName`

`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.FieldWidth`
Width (in pixels) of the field window
Default value: `100`
Saved in: `General.SessionFileName`

`General.FieldHeight`
Height (in pixels) of the field window
Default value: `100`
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.QuadricSubdivisions`
Number of subdivisions used to draw points or lines as spheres or cylinders
Default value: `8`
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 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 of small axes (use negative values for right alignment)
  Default value: `-60`
  Saved in: `General.OptionsFileName`

`General.SmallAxesPositionY`
: Y position of small axes (use negative values for bottom alignment)
  Default value: `-40`
  Saved in: `General.OptionsFileName`

`General.SmallAxesSize`
: Size (in pixels) of small axes
  Default value: `30`
  Saved in: `General.OptionsFileName`

`General.SolverPositionX`
Horizontal position (in pixels) of the upper left corner of the solver windows
Default value: `650`
Saved in: `General.SessionFileName`

`General.SolverPositionY`
Vertical position (in pixels) of the upper left corner of the solver windows
Default value: `150`
Saved in: `General.SessionFileName`

`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.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.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: `3`
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.EpsBackground`
: Save image background in PostScript/PDF output
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: `0.5`
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.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.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.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`

3 Geometry module

Gmsh's geometry module provides a simple CAD engine, using a bottom-up (boundary representation) 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.

Compound 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 4 [Mesh module], page 53, for more information about how physical entities affect the way meshes are saved.

3.1 Geometry commands

The next subsections describe all the available geometry commands. These commands can be used anywhere in a Gmsh ASCII text input 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."

3.1.1 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 last *expression* sets the characteristic mesh length at that point. See Section 4.2.1 [Characteristic lengths], page 54, for more information about how this characteristic length information 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.

3.1.2 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.

  (A deprecated synonym for `Ellipse` is `Ellipsis`.)

`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 3.1.3 [Surfaces], page 43.)

`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.

3.1.3 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* `};`
: 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 *expression-list* on the right hand side should the identification number of a single line loop, composed of either three or four elementary lines.

`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 3.1.4 [Volumes], page 44.)

`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.

3.1.4 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).

(A deprecated synonym for `Volume` is `Complex Volume`.)

`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.

3.1.5 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 4.2.2 [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 *expression*s 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 *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; 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 *expression*s giving the X, Y and Z components of the translation vector; the second *expression-list* should contain three *expression*s giving the X, Y and Z direction of the rotation axis; the third *expression-list* should contain three *expression*s 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* `}; ...`

3.1.6 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 (points, lines or surfaces) 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 (points, lines or surfaces) 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 (points, lines or surfaces) 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 (points, lines or surfaces) 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*.

with

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

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

3.1.7 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 3.2 [Geometry options], page 46).

`Delete { Point | Line | Surface | Volume {` *expression-list* `}; ... }`
: Deletes all elementary entities (points, lines, surfaces or volumes) whose identification numbers are given in *expression-list*.

`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 `"*"`).

3.2 Geometry options

Geometry options control the behavior of geometry commands, as well as the way geometrical entities are handled in the graphical user interface. For the signification of the 'Saved in:' field in the following list, see Section 2.7 [General options], page 16.

`Geometry.AutoCoherence`
: Should all duplicate entities be automatically removed?
Default value: `1`
Saved in: `General.OptionsFileName`

`Geometry.CirclePoints`
: Number of points used to draw a circle/ellipse
Default value: `20`
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.HighlightOrphans`
Highlight orphan entities (lines connected to a single surface, etc.)?
Default value: `0`
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.Normals`
Display size of normal vectors (in pixels)
Default value: `0`
Saved in: `General.OptionsFileName`

`Geometry.OCCFixSmallEdges`
: Fix small edges in STEP, IGES and BRep models
Default value: `1`
Saved in: `General.OptionsFileName`

`Geometry.OCCFixSmallFaces`
: Fix small faces in STEP, IGES and BRep models
Default value: `1`
Saved in: `General.OptionsFileName`

`Geometry.OCCSewFaces`
: Sew faces in STEP, IGES and BRep models
Default value: `0`
Saved in: `General.OptionsFileName`

`Geometry.OldCircle`
: Use old circle description (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), 3D spheres (1) or scaled spheres (2)
Default value: `0`
Saved in: `General.OptionsFileName`

`Geometry.ScalingFactor`
Global geometry scaling factor
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.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`

4 Mesh module

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

The 2D *unstructured* algorithms generate triangles or both triangles and quadrangles (when `Recombine Surface` is used: see Section 4.2.3 [Miscellaneous mesh commands], page 56). The 3D unstructured algorithms only 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 4.2.2 [Structured grids], page 55, and Section 4.2.3 [Miscellaneous mesh commands], page 56). The 3D structured algorithms generate tetrahedra, hexahedra, prisms and pyramids, depending on the type of the surface meshes they are based on.

4.1 Elementary vs. physical entities

If only elementary geometrical entities are defined (or if the `Mesh.SaveAll` option is set; see Section 4.3 [Mesh options], page 57), the grid produced by the mesh module will be saved "as is". That is, all the elements in the grid will be saved to disk using the identification number of the elementary entities they discretize as their elementary region number (and 0 as their physical region number[1]; Chapter 9 [File formats], page 147). 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 3 [Geometry module], page 41) introduces the notion of "physical" entities (also called "physical groups"). 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 147), 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.

---

[1] 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.

4.2 Mesh commands

The mesh module commands mostly permit to modify the characteristic lengths 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 input ASCII text input files. They have to be given either in the GUI or on the command line (see Chapter 8 [Running Gmsh], page 139, and Section 8.3 [Command-line options], page 140).

4.2.1 Characteristic lengths

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

1. You can specify characteristic lengths at the points of the geometrical model (with the `Point` command: see Section 3.1.1 [Points], page 41). The size of the mesh elements will then be computed by linearly interpolating these characteristic lengths on the initial mesh (see Section 1.2 [Mesh], page 3). 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 4.2.2 [Structured grids], page 55).
2. You can specify characteristic lengths using 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 6.1 [Post-processing commands], page 87, and Chapter 9 [File formats], page 147) 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).

     (Note that you can also load a background mesh directly from the command line using the `-bgm` option (see Section 8.3 [Command-line options], page 140), 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
   - ...

   All target element sizes specified by fields can also be constrained by the characteristic lengths defined in the geometrical model if the `Mesh.ConstrainedBackgroundMesh` option is set.

Fields are supported by all the algorithms except those based on Netgen.

Here are the mesh commands that are related to the specification of characteristic lengths:

`Characteristic Length {` *expression-list* `} =` *expression*`;`
: Modifies the characteristic length of the points whose identification numbers are listed in *expression-list*. The new value is given by *expression*.

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

4.2.2 Structured grids

`Extrude {` *expression-list* `} {` *extrude-list layers* `}`
: Extrudes both the geometry and the mesh using a translation (see Section 3.1.5 [Extrusions], page 44). 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* < . . . < *hn* <= 1). See Section 7.3 [t3.geo], page 125, 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 3.1.5 [Extrusions], page 44). 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 3.1.5 [Extrusions], page 44). The *layers* option is defined as above.

`Extrude { Surface {` *expression-list* `};` *layers* `}`

Extrudes a boundary layer along the normals of the specified surfaces.

`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 characteristic length prescription—see Section 4.2.1 [Characteristic lengths], page 54). 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.

(A deprecated synonym for `Progression` is `Power`.)

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

Selects the surface *expression* to be meshed with the 2D transfinite algorithm. The *expression-list* should contain the identification numbers of three or four points on the boundary of the surface, defining the corners of the transfinite interpolation. The optional argument specifies the way the triangles are oriented when the mesh is not recombined.

`Transfinite Volume {` *expression* `} = {` *expression-list* `};`

Selects a five- or six-face volume *expression* to be meshed with the 3D transfinite algorithm. The *expression-list* 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.

4.2.3 Miscellaneous

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

`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 4.3 [Mesh options], page 57). 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* (smothing only applies to transfinite meshes at the moment).

4.3 Mesh options

Mesh options control the behavior of mesh commands, as well as the way meshes are displayed in the graphical user interface. For the signification of the 'Saved in:' field in the following list, see Section 2.7 [General options], page 16.

`Mesh.Algorithm`
: 2D mesh algorithm (1=MeshAdapt+Delaunay, 4=MeshAdapt, 5=Delaunay)
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.Algorithm3D`
: 3D mesh algorithm (1=Tetgen+Delaunay, 4=Netgen)
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.AllowSwapAngle`
: Treshold 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.C1Continuity`
Impose C1 continuity to high order meshes (only valid in 2D plane and ElemenOrder = 2)
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.CharacteristicLengthExtendFromBoundary`
Extend characteristic lengths from the boundaries inside the surface/volume
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.CharacteristicLengthFactor`
Factor applied to all characteristic lengths
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.CharacteristicLengthMin`
Minimum characteristic length
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.CharacteristicLengthMax`
Maximum characteristic length
Default value: `1e+22`
Saved in: `General.OptionsFileName`

`Mesh.CharacteristicLengthFromCurvature`
Compute characteristic lengths from curvature
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.CharacteristicLengthFromPoints`
Compute characteristic lengths from values given at geometry points
Default value: `1`
Saved in: `General.OptionsFileName`

`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.CutPlane`
Enable mesh cut plane
Default value: `0`
Saved in: `-`

`Mesh.CutPlaneDrawIntersect`
Draw only the volume elements that intersect with the cut plane
Default value: `0`
Saved in: `-`

`Mesh.CutPlaneOnlyVolume`
Cut only the volume elements
Default value: `0`
Saved in: `-`

`Mesh.CutPlaneA`
First cut plane equation coefficient ('A' in 'AX+BY+CZ+D=0')
Default value: `1`
Saved in: `-`

`Mesh.CutPlaneB`
Second cut plane equation coefficient ('B' in 'AX+BY+CZ+D=0')
Default value: `0`
Saved in: `-`

`Mesh.CutPlaneC`
Third cut plane equation coefficient ('C' in 'AX+BY+CZ+D=0')
Default value: `0`
Saved in: `-`

`Mesh.CutPlaneD`
Fourth cut plane equation coefficient ('D' in 'AX+BY+CZ+D=0')
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, 19=vrml, 27=stl, 30=mesh, 31=bdf, 32=cgns, 33=med)

Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.Hexahedra`
Display mesh hexahedra?
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.LabelsFrequency`
Labels display frequency?
Default value: `100`
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.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.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.MshBinary`
Write MSH files in binary format?
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.MshFileVersion`
Version of the MSH file format to use
Default value: `2`
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.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.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.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 2D and 3D meshing algorithm (test other values when the algorithm fails)
Default value: `1e-09`
Saved in: `General.OptionsFileName`

`Mesh.RefineSteps`
Number of refinement steps in the MeshAdapt-based 2D algorithms
Default value: `10`
Saved in: `General.OptionsFileName`

`Mesh.RecombineAlgo`
Recombine algorithm (1=mixed triangles-quadrangles, 2=all quadrangles)
Default value: `1`
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.SaveGroupsOfNodes`
: Save groups of nodes for each physical line and surface (UNV mesh format only)
Default value: `0`
Saved in: `-`

`Mesh.ScalingFactor`
: Global scaling factor applied to the saved mesh
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.SecondOrderIncomplete`
: Create incomplete second order elements? (8-node quads, 20-node hexas, etc.)
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.SecondOrderLinear`
: Should second order vertices simply be created by linear interpolation?
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.Smoothing`
: Number of smoothing steps applied to the final mesh
Default value: `1`
Saved in: `General.OptionsFileName`

`Mesh.SmoothInternalEdges`
: Number of smoothing steps of internal edges for high order meshes
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.SmoothNormals`
: Smooth the mesh normals?
Default value: `0`
Saved in: `General.OptionsFileName`

`Mesh.StlBinary`
: Save STL files in binary format?
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.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.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.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: `{255,160,0}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Two`
Color 2 in color carousel
Default value: `{255,200,0}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Three`
Color 3 in color carousel
Default value: `{255,240,0}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Four`
Color 4 in color carousel
Default value: `{228,255,0}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Five`
Color 5 in color carousel
Default value: `{188,255,0}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Six`
Color 6 in color carousel
Default value: `{148,255,0}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Seven`
Color 7 in color carousel
Default value: `{108,255,0}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Eight`
Color 8 in color carousel
Default value: `{68,255,0}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Nine`
Color 9 in color carousel
Default value: `{0,255,52}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Ten`
Color 10 in color carousel
Default value: `{0,255,132}`
Saved in: `General.OptionsFileName`

`Mesh.Color.Eleven`
: Color 11 in color carousel
  Default value: `{0,255,192}`
  Saved in: `General.OptionsFileName`

`Mesh.Color.Twelve`
: Color 12 in color carousel
  Default value: `{0,216,255}`
  Saved in: `General.OptionsFileName`

`Mesh.Color.Thirteen`
: Color 13 in color carousel
  Default value: `{0,176,255}`
  Saved in: `General.OptionsFileName`

`Mesh.Color.Fourteen`
: Color 14 in color carousel
  Default value: `{0,116,255}`
  Saved in: `General.OptionsFileName`

`Mesh.Color.Fifteen`
: Color 15 in color carousel
  Default value: `{0,76,255}`
  Saved in: `General.OptionsFileName`

`Mesh.Color.Sixteen`
: Color 16 in color carousel
  Default value: `{24,0,255}`
  Saved in: `General.OptionsFileName`

`Mesh.Color.Seventeen`
: Color 17 in color carousel
  Default value: `{84,0,255}`
  Saved in: `General.OptionsFileName`

`Mesh.Color.Eighteen`
: Color 18 in color carousel
  Default value: `{104,0,255}`
  Saved in: `General.OptionsFileName`

`Mesh.Color.Nineteen`
: Color 19 in color carousel
  Default value: `{184,0,255}`
  Saved in: `General.OptionsFileName`

5 Solver module

Five external solvers can be interfaced simultaneously with Gmsh.

If you just want to start a solver from the solver module, with no further interactions between the solver and Gmsh, just edit the options relative to one of the five available solvers (e.g., `Solver.Name0`, `Solver.Executable0`, ...; see Section 5.1 [Solver options], page 69), and set the corresponding "client-server" option to zero (e.g., `Solver.ClientServer0 = 0`). This doesn't require any modification to be made to the solver.

If you want the solver to interact with Gmsh (for error messages, option definitions, post-processing, etc.), you need to link your solver with the '`GmshClient.c`' file and add the appropriate function calls inside your program. You can then proceed as in the previous case, but this time you should set the client-server option to 1 (e.g., `Solver.ClientServer0 = 1`), so that Gmsh and the solver can communicate through a Unix socket. See Section 5.2 [Solver example], page 83, for an example of how to interface a C++ solver. Bindings for solvers written in other languages (C, Perl and Python) are available in the source distribution.

5.1 Solver options

`Solver.SocketName`
: Name of socket (TCP/IP if it contains the ':' character, UNIX otherwise)
Default value: `".gmshsock"`
Saved in: `General.OptionsFileName`

`Solver.Name0`
: Name of solver 0
Default value: `"GetDP"`
Saved in: `General.OptionsFileName`

`Solver.Help0`
: Help string for solver 0
Default value: `"A General environment for the treatment of Discrete Problems. Copyright (C) 1997-2008 Patrick Dular and Christophe Geuzaine. Visit http://www.geuz.org/getdp/ for more info"`
Saved in: `General.OptionsFileName`

`Solver.Executable0`
: System command to launch solver 0 (should not contain the '&' character)
Default value: `"getdp"`
Saved in: `General.OptionsFileName`

`Solver.Extension0`
: Default file name extension for solver 0
Default value: `".pro"`
Saved in: `General.OptionsFileName`

`Solver.MeshName0`
: Default mesh file name for solver 0
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.MeshCommand0`
: Command used to specify the mesh file for solver 0
  Default value: `"-msh %s"`
  Saved in: `General.OptionsFileName`

`Solver.SocketCommand0`
: Command to specify the socket to solver 0
  Default value: `"-socket %s"`
  Saved in: `General.OptionsFileName`

`Solver.NameCommand0`
: Command to specify the problem name to solver 0
  Default value: `"%s"`
  Saved in: `General.OptionsFileName`

`Solver.OptionCommand0`
: Command to get options from solver 0
  Default value: `""`
  Saved in: `General.OptionsFileName`

`Solver.FirstOption0`
: Label of first option for solver 0
  Default value: `"Resolution"`
  Saved in: `General.OptionsFileName`

`Solver.SecondOption0`
: Label of second option for solver 0
  Default value: `"PostOperation"`
  Saved in: `General.OptionsFileName`

`Solver.ThirdOption0`
: Label of third option for solver 0
  Default value: `""`
  Saved in: `General.OptionsFileName`

`Solver.FourthOption0`
: Label of fourth option for solver 0
  Default value: `""`
  Saved in: `General.OptionsFileName`

`Solver.FifthOption0`
: Label of fifth option for solver 0
  Default value: `""`
  Saved in: `General.OptionsFileName`

`Solver.FirstButton0`
: Label of first button for solver 0
  Default value: `"Pre"`
  Saved in: `General.OptionsFileName`

`Solver.FirstButtonCommand0`
Command associated with the first button for solver 0
Default value: `"-pre %s"`
Saved in: `General.OptionsFileName`

`Solver.SecondButton0`
Label of second button for solver 0
Default value: `"Cal"`
Saved in: `General.OptionsFileName`

`Solver.SecondButtonCommand0`
Command associated with the second button for solver 0
Default value: `"-cal"`
Saved in: `General.OptionsFileName`

`Solver.ThirdButton0`
Label of third button for solver 0
Default value: `"Pos"`
Saved in: `General.OptionsFileName`

`Solver.ThirdButtonCommand0`
Command associated with the third button for solver 0
Default value: `"-pos %s"`
Saved in: `General.OptionsFileName`

`Solver.FourthButton0`
Label of fourth button for solver 0
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FourthButtonCommand0`
Command associated with the fourth button for solver 0
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FifthButton0`
Label of fifth button for solver 0
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FifthButtonCommand0`
Command associated with the fifth button for solver 0
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.Name1`
Name of solver 1
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.Help1`
: Help string for solver 1
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.Executable1`
: System command to launch solver 1 (should not contain the '&' character)
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.Extension1`
: Default file name extension for solver 1
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.MeshName1`
: Default mesh file name for solver 1
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.MeshCommand1`
: Command used to specify the mesh file for solver 1
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.SocketCommand1`
: Command to specify the socket to solver 1
Default value: `"-socket %s"`
Saved in: `General.OptionsFileName`

`Solver.NameCommand1`
: Command to specify the problem name to solver 1
Default value: `"%s"`
Saved in: `General.OptionsFileName`

`Solver.OptionCommand1`
: Command to get options from solver 1
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FirstOption1`
: Label of first option for solver 1
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.SecondOption1`
: Label of second option for solver 1
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.ThirdOption1`
: Label of third option for solver 1
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FourthOption1`
: Label of fourth option for solver 1
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FifthOption1`
: Label of fifth option for solver 1
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FirstButton1`
: Label of first button for solver 1
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FirstButtonCommand1`
: Command associated with the first button for solver 1
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.SecondButton1`
: Label of second button for solver 1
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.SecondButtonCommand1`
: Command associated with the second button for solver 1
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.ThirdButton1`
: Label of third button for solver 1
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.ThirdButtonCommand1`
: Command associated with the third button for solver 1
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FourthButton1`
: Label of fourth button for solver 1
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FourthButtonCommand1`
Command associated with the fourth button for solver 1
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FifthButton1`
Label of fifth button for solver 1
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FifthButtonCommand1`
Command associated with the fifth button for solver 1
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.Name2`
Name of solver 2
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.Help2`
Help string for solver 2
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.Executable2`
System command to launch solver 2 (should not contain the '&' character)
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.Extension2`
Default file name extension for solver 2
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.MeshName2`
Default mesh file name for solver 2
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.MeshCommand2`
Command used to specify the mesh file for solver 2
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.SocketCommand2`
Command to specify the socket to solver 2
Default value: `"-socket %s"`
Saved in: `General.OptionsFileName`

`Solver.NameCommand2`
Command to specify the problem name to solver 2
Default value: `"%s"`
Saved in: `General.OptionsFileName`

`Solver.OptionCommand2`
Command to get options from solver 2
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FirstOption2`
Label of first option for solver 2
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.SecondOption2`
Label of second option for solver 2
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.ThirdOption2`
Label of third option for solver 2
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FourthOption2`
Label of fourth option for solver 2
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FifthOption2`
Label of fifth option for solver 2
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FirstButton2`
Label of first button for solver 2
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FirstButtonCommand2`
Command associated with the first button for solver 2
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.SecondButton2`
Label of second button for solver 2
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.SecondButtonCommand2`
Command associated with the second button for solver 2
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.ThirdButton2`
Label of third button for solver 2
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.ThirdButtonCommand2`
Command associated with the third button for solver 2
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FourthButton2`
Label of fourth button for solver 2
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FourthButtonCommand2`
Command associated with the fourth button for solver 2
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FifthButton2`
Label of fifth button for solver 2
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FifthButtonCommand2`
Command associated with the fifth button for solver 2
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.Name3`
Name of solver 3
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.Help3`
Help string for solver 3
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.Executable3`
System command to launch solver 3 (should not contain the ‘&’ character)
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.Extension3`
Default file name extension for solver 3
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.MeshName3`
Default mesh file name for solver 3
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.MeshCommand3`
Command used to specify the mesh file for solver 3
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.SocketCommand3`
Command to specify the socket to solver 3
Default value: `"-socket %s"`
Saved in: `General.OptionsFileName`

`Solver.NameCommand3`
Command to specify the problem name to solver 3
Default value: `"%s"`
Saved in: `General.OptionsFileName`

`Solver.OptionCommand3`
Command to get options from solver 3
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FirstOption3`
Label of first option for solver 3
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.SecondOption3`
Label of second option for solver 3
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.ThirdOption3`
Label of third option for solver 3
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FourthOption3`
Label of fourth option for solver 3
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FifthOption3`
Label of fifth option for solver 3
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FirstButton3`
Label of first button for solver 3
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FirstButtonCommand3`
Command associated with the first button for solver 3
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.SecondButton3`
Label of second button for solver 3
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.SecondButtonCommand3`
Command associated with the second button for solver 3
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.ThirdButton3`
Label of third button for solver 3
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.ThirdButtonCommand3`
Command associated with the third button for solver 3
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FourthButton3`
Label of fourth button for solver 3
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FourthButtonCommand3`
Command associated with the fourth button for solver 3
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FifthButton3`
Label of fifth button for solver 3
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FifthButtonCommand3`
Command associated with the fifth button for solver 3
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.Name4`
Name of solver 4
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.Help4`
Help string for solver 4
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.Executable4`
System command to launch solver 4 (should not contain the '&' character)
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.Extension4`
Default file name extension for solver 4
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.MeshName4`
Default mesh file name for solver 4
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.MeshCommand4`
Command used to specify the mesh file for solver 4
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.SocketCommand4`
Command to specify the socket to solver 4
Default value: `"-socket %s"`
Saved in: `General.OptionsFileName`

`Solver.NameCommand4`
Command to specify the problem name to solver 4
Default value: `"%s"`
Saved in: `General.OptionsFileName`

`Solver.OptionCommand4`
Command to get options from solver 4
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FirstOption4`
: Label of first option for solver 4
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.SecondOption4`
: Label of second option for solver 4
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.ThirdOption4`
: Label of third option for solver 4
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FourthOption4`
: Label of fourth option for solver 4
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FifthOption4`
: Label of fifth option for solver 4
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FirstButton4`
: Label of first button for solver 4
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FirstButtonCommand4`
: Command associated with the first button for solver 4
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.SecondButton4`
: Label of second button for solver 4
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.SecondButtonCommand4`
: Command associated with the second button for solver 4
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.ThirdButton4`
: Label of third button for solver 4
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.ThirdButtonCommand4`
Command associated with the third button for solver 4
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FourthButton4`
Label of fourth button for solver 4
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FourthButtonCommand4`
Command associated with the fourth button for solver 4
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FifthButton4`
Label of fifth button for solver 4
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.FifthButtonCommand4`
Command associated with the fifth button for solver 4
Default value: `""`
Saved in: `General.OptionsFileName`

`Solver.AlwaysListen`
Always listen to incoming connection requests?
Default value: `0`
Saved in: `General.OptionsFileName`

`Solver.ClientServer0`
Connect solver 0 to the Gmsh server
Default value: `1`
Saved in: `General.OptionsFileName`

`Solver.ClientServer1`
Connect solver 1 to the Gmsh server
Default value: `0`
Saved in: `General.OptionsFileName`

`Solver.ClientServer2`
Connect solver 2 to the Gmsh server
Default value: `0`
Saved in: `General.OptionsFileName`

`Solver.ClientServer3`
Connect solver 3 to the Gmsh server
Default value: `0`
Saved in: `General.OptionsFileName`

`Solver.ClientServer4`
Connect solver 4 to the Gmsh server
Default value: `0`
Saved in: `General.OptionsFileName`

`Solver.MaximumDelay`
Maximum delay (in seconds) allowed for solver response
Default value: `4`
Saved in: `General.OptionsFileName`

`Solver.MergeViews0`
Automatically merge any post-processing view created by solver 0
Default value: `1`
Saved in: `General.OptionsFileName`

`Solver.MergeViews1`
Automatically merge any post-processing view created by solver 1
Default value: `1`
Saved in: `General.OptionsFileName`

`Solver.MergeViews2`
Automatically merge any post-processing view created by solver 2
Default value: `1`
Saved in: `General.OptionsFileName`

`Solver.MergeViews3`
Automatically merge any post-processing view created by solver 3
Default value: `1`
Saved in: `General.OptionsFileName`

`Solver.MergeViews4`
Automatically merge any post-processing view created by solver 4
Default value: `1`
Saved in: `General.OptionsFileName`

`Solver.Plugins`
Enable default solver plugins?
Default value: `0`
Saved in: `General.OptionsFileName`

`Solver.PopupMessages0`
Automatically display messages produced by solver 0
Default value: `1`
Saved in: `General.OptionsFileName`

`Solver.PopupMessages1`
Automatically display messages produced by solver 1
Default value: `1`
Saved in: `General.OptionsFileName`

`Solver.PopupMessages2`
Automatically display messages produced by solver 2
Default value: `1`
Saved in: `General.OptionsFileName`

`Solver.PopupMessages3`
Automatically display messages produced by solver 3
Default value: `1`
Saved in: `General.OptionsFileName`

`Solver.PopupMessages4`
Automatically display messages produced by solver 4
Default value: `1`
Saved in: `General.OptionsFileName`

5.2 Solver example

Here is a small example of how to interface a C++ solver with Gmsh. The following listing reproduces the '`utils/solvers/c++/solver.cpp`' file from the Gmsh source distribution (C, Perl and Python examples are also available).

```
// $Id: solver.cpp,v 1.10 2007-09-04 13:47:08 remacle Exp $
//
// Copyright (C) 1997-2005 C. Geuzaine, J.-F. Remacle
//
// Permission is hereby granted, free of charge, to any person
// obtaining a copy of this software and associated documentation
// files (the "Software"), to deal in the Software without
// restriction, including without limitation the rights to use, copy,
// modify, merge, publish, distribute, and/or sell copies of the
// Software, and to permit persons to whom the Software is furnished
// to do so, provided that the above copyright notice(s) and this
// permission notice appear in all copies of the Software and that
// both the above copyright notice(s) and this permission notice
// appear in supporting documentation.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
// EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
// NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE
// COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR
// ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY
// DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS,
// WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS
// ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE
// OF THIS SOFTWARE.
//
```

```
// Please report all bugs and problems to <gmsh@geuz.org>.

// This file contains a dummy client solver for Gmsh. It does not
// solve anything, but shows how to program your own solver to interact
// with the Gmsh solver module.
//
// To compile this solver, type something like:
//
// g++ solver.cpp -o solver.exe
//
// To run it, merge the contents of the file solver.opt into your
// default Gmsh option file, or launch Gmsh with the command:
//
// gmsh -option solver.opt
//
// You will then see a new button labeled "My C++ solver" in Gmsh's
// solver menu.

#include <math.h>
#include "GmshClient.h"

typedef enum { send_options, run_code } action;

int main(int argc, char *argv[])
{
  action what_to_do = run_code;
  char *name = NULL, *option = NULL, *socket = NULL;

  // parse command line

  int i = 0;
  while(i < argc) {
    if(argv[i][0] == '-') {
      if(!strcmp(argv[i] + 1, "socket")) {
        i++;
        if(argv[i]) socket = argv[i++];
      }
      else if(!strcmp(argv[i] + 1, "options")) {
        i++;
        what_to_do = send_options;
      }
      else if(!strcmp(argv[i] + 1, "run")) {
        i++;
        what_to_do = run_code;
        if(argv[i]) option = argv[i++];
      }
    }
```

```
    else
      name = argv[i++];
  }

  if(!socket) {
    printf("No socket specified: running non-interactively...\n");
    exit(1);
  }

  // connect to Gmsh

  GmshClient client;
  if(client.Connect(socket) < 0){
    printf("Unable to connect to Gmsh\n");
    exit(1);
  }

  client.Start();

  if(what_to_do == send_options) {
    // send the available options for this computation
    client.Option(1, "FormulationH");
    client.Option(1, "ConvTest");
    client.Option(1, "Blablabli");
  }
  else if(what_to_do == run_code){
    // do the computation and merge some views
    for(int i = 0; i < 10; i++){
      client.Info("Computing curve...");
      // fake computation for 500ms:
#if !defined(WIN32) || defined(__CYGWIN__)
      usleep(500 * 1000);
#else
      Sleep(500);
#endif
      client.Info("Done computing curve");
      FILE *file = fopen("solver.pos", "w");
      if(!file)
        client.Error("Unable to open output file");
      else {
        fprintf(file, "View.Type = 2;\n");
        fprintf(file, "View.Axes = 3;\n");
        fprintf(file, "Delete View[0];\n");
        fprintf(file, "View \"%s\"{\n", option);
        for(int j = 0; j < 100; j++)
          fprintf(file, "SP(%d,0,0){%g};\n", j,sin(j*i*M_PI/10.));
        fprintf(file, "};\n");
```

```
        fclose(file);
        client.MergeFile("solver.pos");
      }
    }
    client.Info("Done!");
  }

  client.Stop();
  client.Disconnect();
}
```

To define the above solver as the second external solver in Gmsh, you then need to define the following options (either merge them in your Gmsh option file, or use the `-option` command-line option—see Section 8.3 [Command-line options], page 140):

```
Solver.Name1 = "My C++ Solver";
Solver.Executable1 = "solver.exe";
Solver.OptionCommand1 = "-options";
Solver.FirstOption1 = "My options";
Solver.FirstButton1 = "Run !";
Solver.FirstButtonCommand1 = "-run %s";
Solver.ClientServer1 = 1;
Solver.MergeViews1 = 1;
Solver.PopupMessages1 = 1;
```

6 Post-processing module

Gmsh's post-processing module can handle multiple scalar, vector or tensor data sets along with the geometry and the mesh. The data sets should be given in one of Gmsh's post-processing file formats described in Chapter 9 [File formats], page 147. 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. To display other (combinations of) components, use `Plugin(Extract)`: see Section 6.2 [Post-processing plugins], page 91.)

In Gmsh's jargon, each data set is called a "view", and can arbitrarily mix all types of elements and fields. 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 6.3 [Post-processing options], page 104).

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 7.8 [t8.geo], page 134, and Section 7.9 [t9.geo], page 137).

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 are available in the '`tutorial`' directory of Gmsh's distribution ('`.pos`' files).

6.1 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 6.2 [Post-processing plugins], page 91.

`Plugin (`*string*`) .` *string* `=` *expression* `|` *char-expression*`;`

Sets an option for a given plugin. See Section 6.2 [Post-processing plugins], page 91, for a list of default plugins and Section 7.9 [t9.geo], page 137, 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 7.4 [t4.geo], page 127), 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 147, or use one of the standard formats like MED.

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

```
View "string" {
  < TIME { expression-list }; >
  type ( list-of-coords ) { list-of-values };
  ...
};
```

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*
`2nd order scalar line`	`SL2`	`9`	`3 *` *nb-time-steps*
`2nd order vector line`	`VL2`	`9`	`9 *` *nb-time-steps*
`2nd order tensor line`	`TL2`	`9`	`27 *` *nb-time-steps*
`2nd order scalar triangle`	`ST2`	`18`	`6 *` *nb-time-steps*
`2nd order vector triangle`	`VT2`	`18`	`18 *` *nb-time-steps*
`2nd order tensor triangle`	`TT2`	`18`	`54 *` *nb-time-steps*
`2nd order scalar quadrangle`	`SQ2`	`27`	`9 *` *nb-time-steps*
`2nd order vector quadrangle`	`VQ2`	`27`	`27 *` *nb-time-steps*
`2nd order tensor quadrangle`	`TQ2`	`27`	`81 *` *nb-time-steps*
`2nd order scalar tetrahedron`	`SS2`	`30`	`10 *` *nb-time-steps*
`2nd order vector tetrahedron`	`VS2`	`30`	`30 *` *nb-time-steps*
`2nd order tensor tetrahedron`	`TS2`	`30`	`90 *` *nb-time-steps*
`2nd order scalar hexahedron`	`SH2`	`81`	`27 *` *nb-time-steps*
`2nd order vector hexahedron`	`VH2`	`81`	`81 *` *nb-time-steps*
`2nd order tensor hexahedron`	`TH2`	`81`	`243*` *nb-time-steps*
`2nd order scalar prism`	`SI2`	`54`	`18 *` *nb-time-steps*
`2nd order vector prism`	`VI2`	`54`	`54 *` *nb-time-steps*
`2nd order tensor prism`	`TI2`	`54`	`162*` *nb-time-steps*
`2nd order scalar pyramid`	`SY2`	`42`	`14 *` *nb-time-steps*
`2nd order vector pyramid`	`VY2`	`42`	`42 *` *nb-time-steps*
`2nd order tensor pyramid`	`TY2`	`42`	`126*` *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 152. For second order elements, the first order nodes are given first, followed by the nodes associated with the edges, followed by the nodes associated with the quadrangular faces (if any), followed by the nodes associated with the volume (if any). The ordering of these additional nodes follows the ordering of the edges and quadrangular faces given in Section 9.3 [Node ordering], page 152.

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 *expression*s 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-expression*s.

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.

6.2 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 6.3 [Post-processing options], page 104) 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 graphical user interface 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(Annotate)`

Plugin(Annotate) adds the text string 'Text', in font 'Font' and size 'FontSize', in the view 'iView'. 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'). The string is aligned according to 'Align'. If 'iView' < 0, the plugin is run on the current view.

Plugin(Annotate) is executed in-place.

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`

`ThereD` Default value: `0`

`FontSize` Default value: `14`

`iView` Default value: `-1`

`Plugin(Curl)`

Plugin(Curl) computes the curl of the field in the view 'iView'. If 'iView' < 0, the plugin is run on the current view.

Plugin(Curl) creates one new view.

Numeric options:

`iView` Default value: `-1`

`Plugin(CutGrid)`

Plugin(CutGrid) cuts the view 'iView' 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 'nPointsU' and 'nPointsV'. If 'ConnectPoints' is zero, the plugin creates points; otherwise, the plugin generates quadrangles, lines or points depending on the values of 'nPointsU' and 'nPointsV'. If 'iView' < 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`

`nPointsU` Default value: `20`

`nPointsV` Default value: `20`

`ConnectPoints`
Default value: `1`

`iView` Default value: `-1`

`Plugin(CutMap)`

Plugin(CutMap) extracts the isosurface of value 'A' from the view 'iView' and draws the 'dTimeStep'-th value of the view 'dView' on the isosurface. If 'iView' < 0, the plugin is run on the current view. If 'dTimeStep' < 0, the plugin uses, for each time step in 'iView', the corresponding time step in 'dView'. If 'dView' < 0, the plugin uses 'iView' as the value source. If 'ExtractVolume' is nonzero, the plugin extracts the isovolume with values greater (if 'ExtractVolume' > 0) or smaller (if 'ExtractVolume' < 0) than the isosurface 'A'.

Plugin(CutMap) creates as many views as there are time steps in 'iView'.

Numeric options:

`A` Default value: `0`

`dTimeStep`
Default value: `-1`

`dView` Default value: `-1`

`ExtractVolume`
Default value: `0`

`RecurLevel`
Default value: `4`

`TargetError`
Default value: `0`

`iView` Default value: `-1`

`Plugin(CutParametric)`

Plugin(CutParametric) cuts the view 'iView' with the parametric function ('X'(u), 'Y'(u), 'Z'(u)), using 'nPointsU' values of the parameter u in ['MinU', 'MaxU']. If 'ConnectPoints' is set, the plugin creates line elements; otherwise, the plugin generates points. If 'iView' < 0, the plugin is run on the current view.

Plugin(CutParametric) creates one new view.

String options:

`X` Default value: `"0 + 1 * Cos(u)"`

`Y` Default value: `"0 + 1 * Sin(u)"`

`Z` Default value: `"0"`

Numeric options:

`MinU` Default value: `0`

`MaxU` Default value: `6.2832`

`nPointsU` Default value: `360`

`ConnectPoints`
Default value: `0`

`iView` Default value: `-1`

`Plugin(CutPlane)`
Plugin(CutPlane) cuts the view 'iView' 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 'iView' < 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`

`iView` Default value: `-1`

`Plugin(CutSphere)`
Plugin(CutSphere) cuts the view 'iView' 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 'iView' < 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`

`iView` Default value: `-1`

`Plugin(Divergence)`

Plugin(Divergence) computes the divergence of the field in the view 'iView'. If 'iView' < 0, the plugin is run on the current view.

Plugin(Divergence) creates one new view.

Numeric options:

`iView` Default value: `-1`

`Plugin(Eigenvalues)`

Plugin(Eigenvalues) computes the three real eigenvalues of each tensor in the view 'iView'. If 'iView' < 0, the plugin is run on the current view.

Plugin(Eigenvalues) creates three new scalar views.

Numeric options:

`iView` Default value: `-1`

`Plugin(Eigenvectors)`

Plugin(Eigenvectors) computes the three (right) eigenvectors of each tensor in the view 'iView' 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 'iView' < 0, the plugin is run on the current view.

Plugin(Eigenvectors) creates three new vector views.

Numeric options:

`ScaleByEigenvalues`
Default value: `1`

`iView` Default value: `-1`

`Plugin(Evaluate)`

Plugin(Evaluate) sets the 'Component'-th component of the 'TimeStep'-th time step in the view 'iView' 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 'iView' at the 'TimeStep'-th time step;

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

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

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

If 'TimeStep' < 0, the plugin automatically loops over all the time steps in 'iView' and evaluates 'Expression' for each one. If 'ExternalTimeStep' < 0, the plugin uses 'TimeStep' instead. If 'Component' < 0, the plugin automatically loops over all the components in the view and evaluates 'Expression' for each one. If 'iView' < 0, the plugin is run on the current view. If 'ExternalView' < 0, the plugin uses 'iView' instead.

Plugin(Evaluate) is executed in-place.

String options:

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

Numeric options:

`Component`
: Default value: `-1`

`TimeStep`
: Default value: `-1`

`ExternalView`
: Default value: `-1`

`ExternalTimeStep`
: Default value: `-1`

`iView`
: Default value: `-1`

`Plugin(Extract)`

Plugin(Extract) extracts a combination of components from the 'TimeStep'th time step in the view 'iView'. 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 of the field, and 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 'iView' < 0, the plugin is run on the current view.

Plugin(Extract) 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

`iView` Default value: -1

`Plugin(ExtractElements)`

Plugin(ExtractElements) extracts the elements from the view 'iView' whose 'TimeStep'-th values (averaged by element) are comprised between 'MinVal' and 'MaxVal'. If 'iView' < 0, the plugin is run on the current view.

Plugin(ExtractElements) creates one new view.

Numeric options:

`MinVal` Default value: 0

`MaxVal` Default value: 1

`TimeStep` Default value: 0

`iView` Default value: -1

`Plugin(FieldView)`

Plugin(FieldView) evaluate a field on the choosen view.

Numeric options:

`Component`
: Default value: -1

`iView` Default value: -1

`iField` Default value: -1

`Plugin(GSHHS)`

Plugin(GSHHS) import GSHHS data.

String options:

`InFileName`
: Default value: "gshhs_c.b"

`OutFileName`
: Default value: "earth.geo"

`Format` Default value: `"gshhs"`

`Coordinate`
Default value: `"cartesian"`

Numeric options:

`iField` Default value: `-1`

`UTMZone` Default value: `0`

`UTMEquatorialRadius`
Default value: `6.37814e+06`

`UTMPolarRadius`
Default value: `6.35675e+06`

`UTMScale` Default value: `1`

`UTMShiftX`
Default value: `0`

`UTMShiftY`
Default value: `0`

`WritePolarSphere`
Default value: `1`

`Plugin(Gradient)`

Plugin(Gradient) computes the gradient of the field in the view ‘iView’. If ‘iView’ < 0, the plugin is run on the current view.

Plugin(Gradient) creates one new view.

Numeric options:

`iView` Default value: `-1`

`Plugin(HarmonicToTime)`

Plugin(HarmonicToTime) takes the values in the time steps ‘RealPart’ and ‘ImaginaryPart’ of the view ‘iView’, and creates a new view containing (‘iView’[‘RealPart’] * cos(p) - ‘iView’[‘ImaginaryPart’] * sin(p)), with p = 2*Pi*k/‘nSteps’, k = 0, ..., ‘nSteps’-1. If ‘iView’ < 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`

`nSteps` Default value: `20`

`iView` Default value: `-1`

`Plugin(Integrate)`

Plugin(Integrate) integrates scalar fields over all the elements in the view 'iView', as well as the circulation/flux of vector fields over line/surface elements. If 'iView' < 0, the plugin is run on the current view.

Plugin(Integrate) creates one new view.

Numeric options:

`iView` 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 'iView' contains tensor elements, the plugin directly uses the tensors as the values of the velocity gradient tensor; if 'iView' contains vector elements, the plugin uses them as the velocities from which to derive the velocity gradient tensor. If 'iView' < 0, the plugin is run on the current view.

Plugin(Lambda2) creates one new view.

Numeric options:

`Eigenvalue`
Default value: `2`

`iView` Default value: `-1`

`Plugin(MakeSimplex)`

Plugin(MakeSimplex) decomposes all non- simplectic elements (quadrangles, prisms, hexahedra, pyramids) in the view 'iView' into simplices (triangles, tetrahedra). If 'iView' < 0, the plugin is run on the current view.

Plugin(MakeSimplex) is executed in-place.

Numeric options:

`iView` Default value: `-1`

`Plugin(ModulusPhase)`

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

`iView` Default value: `-1`

`Plugin(Probe)`

Plugin(Probe) gets the value of the view 'iView' at the point ('X','Y','Z'). If 'iView' < 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`

`iView` Default value: `-1`

`Plugin(Remove)`

Plugin(Remove) removes the marked items from the view 'iView'. If 'iView' < 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`

`iView` Default value: `-1`

`Plugin(Skin)`

Plugin(Skin) extracts the skin (the boundary) of the view 'iView'. If 'iView' < 0, the plugin is run on the current view.

Plugin(Skin) creates one new view.

Numeric options:

`iView` Default value: `-1`

`Plugin(Smooth)`

Plugin(Smooth) averages the values at the nodes of the scalar view ‘iView’. If ‘iView’ < 0, the plugin is run on the current view.

Plugin(Smooth) is executed in-place.

Numeric options:

`iView` Default value: `-1`

`Plugin(SphericalRaise)`

Plugin(SphericalRaise) transforms the coordinates of the elements in the view ‘iView’ using the values associated with the ‘TimeStep’-th time step. Instead of elevating the nodes along the X, Y and Z axes as in View[‘iView’].RaiseX, View[‘iView’].RaiseY and View[‘iView’].RaiseZ, 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 ‘iView’ < 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`

`iView` Default value: `-1`

`Plugin(StreamLines)`

Plugin(StreamLines) computes stream lines from the ‘TimeStep’-th time step of a vector view ‘iView’ and optionally interpolates the scalar view ‘dView’ 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 that are to be transported along U and V is set with the options ‘nPointsU’ and ‘nPointsV’. 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 ‘iView’ < 0, the plugin is run on the current view. If ‘TimeStep’ < 0, the plugin tries to compute streamlines of the unsteady flow.

Plugin(StreamLines) creates one new view. This view contains multi-step vector points if ‘dView’ < 0, or single-step scalar lines if ‘dView’ >= 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`

`nPointsU` Default value: `10`

`nPointsV` Default value: `1`

`MaxIter` Default value: `100`

`DT` Default value: `0.1`

`TimeStep` Default value: `0`

`dView` Default value: `-1`

`iView` Default value: `-1`

`Plugin(Transform)`

Plugin(Transform) transforms the homogeneous node coordinates (x,y,z,1) of the elements in the view 'iView' 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 'iView' `<` 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`

`iView` Default value: `-1`

`Plugin(TransformLatLon)`
Plugin(TransformLatLon) transforms the homogeneous node coordinates (x,y,z,1) of the elements in the view 'iView' 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 'iView' < 0, the plugin is run on the current view.

Plugin(TransformLatLon) is executed in-place.

Numeric options:

`iView` Default value: `-1`

`Longitude0`
Default value: `0`

`Plugin(Triangulate)`
Plugin(Triangulate) triangulates the points in the view 'iView', assuming that all the points belong to a surface that can be projected one-to-one onto a plane. If 'iView' < 0, the plugin is run on the current view.

Plugin(Triangulate) creates one new view.

Numeric options:

`iView` Default value: `-1`

`Plugin(Warp)`
Plugin(Warp) transforms the elements in the view 'iView' by adding to their node coordinates the vector field stored in the 'TimeStep'-th time step of the view 'dView', scaled by 'Factor'. If 'dView' < 0, the vector field is taken as the field of surface normals multiplied by the 'TimeStep' value in 'iView'. (The smoothing of the surface normals is controlled by the 'SmoothingAngle' parameter.) If 'iView' < 0, the plugin is run on the current view.

Plugin(Warp) is executed in-place.

Numeric options:

`Factor` Default value: `1`

`TimeStep` Default value: `0`

`SmoothingAngle`
Default value: `180`

`dView` Default value: `-1`

`iView` Default value: `-1`

6.3 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*.

See Section 7.8 [t8.geo], page 134, and Section 7.9 [t9.geo], page 137, for some examples.

`PostProcessing.AnimationDelay`
: Delay (in seconds) between frames in automatic animation mode
Default value: `0.25`
Saved in: `General.OptionsFileName`

`PostProcessing.AnimationCycle`
: Cycle through views instead of time steps in automatic animation mode
Default value: `0`
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=text, 5=mesh)
Default value: `0`
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.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.AngleSmoothNormals`
Threshold angle below which normals are not smoothed
Default value: `30`
Saved in: `General.OptionsFileName`

`View.ArrowHeadRadius`
Relative radius of arrow head
Default value: `0.12`
Saved in: `General.OptionsFileName`

`View.ArrowSize`
Display size of arrows (in pixels)
Default value: `60`
Saved in: `General.OptionsFileName`

`View.ArrowSizeProportional`
Scale the arrows according to the norm of the vector
Default value: `1`
Saved in: `General.OptionsFileName`

`View.ArrowStemLength`
Relative length of arrow stem
Default value: `0.56`
Saved in: `General.OptionsFileName`

`View.ArrowStemRadius`
Relative radius of arrow stem
Default value: `0.02`
Saved in: `General.OptionsFileName`

`View.AutoPosition`
Position the scale or 2D plot automatically
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.)
Default value: `0`
Saved in: `General.OptionsFileName`

`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.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.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: `15`
  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) or 3D spheres (1)
  Default value: `0`
  Saved in: `General.OptionsFileName`

`View.PositionX`
: Horizontal position (in pixels) of the upper left corner of the scale or 2D plot
  Default value: `100`
  Saved in: `General.OptionsFileName`

`View.PositionY`
: Vertical position (in pixels) of the upper left corner of the scale or 2D plot
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.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=value if multi-step, 2=value always, 3=step if multi-step, 4=step always)
Default value: `1`
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.Transform11`
: Element (1,1) of the 3x3 coordinate transformation matrix
Default value: `1`
Saved in: `-`

`View.Transform12`
: Element (1,2) of the 3x3 coordinate transformation matrix
Default value: `0`
Saved in: `-`

`View.Transform13`
: Element (1,3) of the 3x3 coordinate transformation matrix
Default value: `0`
Saved in: `-`

`View.Transform21`
: Element (2,1) of the 3x3 coordinate transformation matrix
Default value: `0`
Saved in: `-`

`View.Transform22`
: Element (2,2) of the 3x3 coordinate transformation matrix
Default value: `1`
Saved in: `-`

`View.Transform23`
: Element (2,3) of the 3x3 coordinate transformation matrix
Default value: `0`
Saved in: `-`

`View.Transform31`
: Element (3,1) of the 3x3 coordinate transformation matrix
Default value: `0`
Saved in: `-`

`View.Transform32`
: Element (3,2) of the 3x3 coordinate transformation matrix
Default value: `0`
Saved in: `-`

`View.Transform33`
: 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)
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.ColorTable`
: Color table used to draw the view
Saved in: `General.OptionsFileName`

7 Tutorial

The nine 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.

This tutorial does not explain the mesh and post-processing file formats: see Chapter 9 [File formats], page 147, for this.

To learn how to run Gmsh on your computer, see Chapter 8 [Running Gmsh], page 139. In addition, interactive tutorials that explain how to use the graphical user interface are available on Gmsh's webpage:

- The first interactive tutorial shows how to create a simple 2D geometry and how to generate a mesh;
- The second shows several methods to visualize data sets on deformed grids;
- The third shows how to use post-processing plugins.

7.1 '`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 = 0.009;

// 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), and specifying their orientation.

// 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.
```

7.2 '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:

tmp[] = Translate {0, 0.1, 0} { Duplicata{ Point{3}; } } ;

// In this case, we assigned the result of the Translate command to a
```

```
// list, so that we can retrieve the number of the newly created point
// and use it to create new lines and a new surface:

Line(7) = {3,tmp[0]};
Line(8) = {tmp[0],5};
Line Loop(10) = {5,-8,-7,3};
Plane Surface(11) = {10};

// Of course, these transformation commands not only apply to points,
// but also to lines and surfaces. We can for example translate a copy
// of surface 6 by 0.12 units along the z axis and define some
// additional lines and surfaces with:

h = 0.12;
Translate {0, 0, h} { Duplicata{ Surface{6}; } }

Line(106) = {1,8};
Line(107) = {2,12};
Line(108) = {3,16};
Line(109) = {4,7};

Line Loop(110) = {1,107,-103,-106}; Plane Surface(111) = {110};
Line Loop(112) = {2,107,104,-108};  Plane Surface(113) = {112};
Line Loop(114) = {3,109,-105,-108}; Plane Surface(115) = {114};
Line Loop(116) = {4,106,-102,-109}; Plane Surface(117) = {116};

// 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:

Surface Loop(118) = {117,-6,111,-113,101,115};
Volume(119) = {118};

// Another way to define a volume is by extruding a surface. The
// following command extrudes the surface 11 along the z axis and
// automatically creates a new volume:

Extrude {0, 0, h} { Surface{11}; }

// All these geometrical transformations automatically generate new
// elementary entities. The following command permits to manually
// assign a characteristic length to some of the new points:

Characteristic Length {tmp[0], 2, 12, 3, 16, 6, 22} = lc * 4;
```

```
// 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 119 and 120
// with a common region number, we finally define a physical
// volume:

Physical Volume (1) = {119,120};
```

7.3 ‘t3.geo’

```
/*********************************************************************
 *
 *  Gmsh tutorial 3
 *
 *  Extruded meshes, 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:

out[] = Extrude { {-2*h,0,0}, {1,0,0} , {0,0.15,0.25} , Pi/2 } {
  Surface{news-1}; Layers{10}; Recombine;
};

// In this last extrusion command we retrieved the volume number
// programatically by saving the output of the command into a
// list. This list will contain the "top" of the extruded surface (in
// out[0]) as well as the newly created volume (in out[1]).

// 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 click on the '?'  button in the status bar of the graphic
// window to see the current values of all options. To save all the
// options in a file, you can use the 'File->Save as->Gmsh options'
// menu. To save the current options as the default options for all
```

```
// future Gmsh sessions, you should use the 'Tools->Options->Save as
// defaults' button.
```

7.4 't4.geo'

```
/*********************************************************************
 *
 *  Gmsh tutorial 4
 *
 *  Built-in functions, holes, strings, mesh color
 *
 *********************************************************************/

// As usual, we start by defining some variables, some points and some
// lines:

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;

ccos = ( -h5*R1 + e2 * Hypot(h5,Hypot(e2,R1)) ) / (h5^2 + e2^2);
ssin = Sqrt(1-ccos^2);

Lc1 = 0.01;
Lc2 = 0.003;

Point(1) = { -e1-e2, 0.0  , 0.0 , Lc1};
Point(2) = { -e1-e2, h1   , 0.0 , Lc1};
Point(3) = { -e3-r , h1   , 0.0 , Lc2};
Point(4) = { -e3-r , h1+r , 0.0 , Lc2};
Point(5) = { -e3   , h1+r , 0.0 , Lc2};
Point(6) = { -e3   , h1+h2, 0.0 , Lc1};
Point(7) = {  e3   , h1+h2, 0.0 , Lc1};
Point(8) = {  e3   , h1+r , 0.0 , Lc2};
Point(9) = {  e3+r , h1+r , 0.0 , Lc2};
Point(10)= {  e3+r , h1   , 0.0 , Lc2};
Point(11)= {  e1+e2, h1   , 0.0 , Lc1};
Point(12)= {  e1+e2, 0.0  , 0.0 , Lc1};
Point(13)= {  e2   , 0.0  , 0.0 , Lc1};

Point(14)= {  R1 / ssin , h5+R1*ccos, 0.0 , Lc2};
Point(15)= {  0.0       , h5        , 0.0 , Lc2};
```

```
Point(16)= { -R1 / ssin , h5+R1*ccos, 0.0 , Lc2};
Point(17)= { -e2        , 0.0       , 0.0 , Lc1};

Point(18)= { -R2  , h1+h3   , 0.0 , Lc2};
Point(19)= { -R2  , h1+h3+h4, 0.0 , Lc2};
Point(20)= {  0.0 , h1+h3+h4, 0.0 , Lc2};
Point(21)= {  R2  , h1+h3+h4, 0.0 , Lc2};
Point(22)= {  R2  , h1+h3   , 0.0 , Lc2};
Point(23)= {  0.0 , h1+h3   , 0.0 , Lc2};

Point(24)= {  0 , h1+h3+h4+R2, 0.0 , Lc2};
Point(25)= {  0 , h1+h3-R2,    0.0 , 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};

// 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" {
  // 10 pixels from the left and 15 pixels from the top of the graphic
  // window:
  T2(10,15,0){StrCat("File created on ", Today)};

  // 10 pixels from the left and 10 pixels from the bottom of the
  // graphic window:
  T2(10,-10,0){"Copyright (C) My Company"};

  // in the model, at (X,Y,Z) = (0.0,0.11,0.0):
  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 }; }
```

7.5 ‘t5.geo’

```
/*********************************************************************
 *
 *  Gmsh tutorial 5
 *
 *  Characteristic lengths, arrays of variables, functions, loops
 *
 *********************************************************************/

// Again, we start be defining some characteristic lengths:

lcar1 = .1;
lcar2 = .0005;
lcar3 = .055;

// If we wanted to change these lengths 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 3,000 nodes and
// 15,000 tetrahedra. With
//
// > gmsh t5.geo -clscale 0.2
//
// the mesh counts approximately 600,000 nodes and 3.6 million
// tetrahedra.

// 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 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 ;

  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):

  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 exterior surface and the five interior loops).
// To reference an array of variables, its identifier is followed by
// '[]':

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 ;
```

7.6 '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";

// Put 20 equidistant points on curve 4
Transfinite Line{4} = 20 ;

// Put 20 points with a refinement toward the extremities on curve 2
Transfinite Line{2} = 20 Using Bump 0.05;

// 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{6} = {1,2,3,4};

// (Note that the list on the right hand side refers to points, not
// curves. The way triangles are generated can be controlled by
// appending "Left", "Right" or "Alternate" after the list.)

// Recombine the triangles into quads
Recombine Surface{6};

// Apply an elliptic smoother to the grid
Mesh.Smoothing = 100;
```

7.7 ‘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];
```

7.8 ‘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
```

7.9 ‘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. CutMap, 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 'CutMap' plugin (which extracts an
// isovalue surface from a 3D scalar view), and run it:

Plugin(CutMap).A = 0.67 ; // iso-value level
Plugin(CutMap).iView = 0 ; // source view is View[0]
Plugin(CutMap).Run ;
```

```
// We also set some options for the ‘CutPlane’ plugin (which computes
// a section of a 3D view), and then run it:

Plugin(CutPlane).A = 0 ;
Plugin(CutPlane).B = 0.2 ;
Plugin(CutPlane).C = 1 ;
Plugin(CutPlane).D = 0 ;
Plugin(CutPlane).Run ;

// Add a title

Plugin(Annotate).Text = "A nice title" ;
// By convention, 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).X = 1.e5;
Plugin(Annotate).Y = 50 ;
Plugin(Annotate).Font = "Times-BoldItalic" ;
Plugin(Annotate).FontSize = 28 ;
Plugin(Annotate).Align = "Center" ;
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;
```

8 Running Gmsh

8.1 Interactive mode

Gmsh's first operating mode is the 'interactive graphical mode'. To launch Gmsh in interactive mode, just click or double-click on the Gmsh icon (Windows and Mac), or type

```
> gmsh
```

at your shell prompt in a terminal (Unix). This will open two windows: the graphic window (with a status bar at the bottom) and the menu window (with a menu bar and some context-dependent buttons).

To open the first tutorial file (see Chapter 7 [Tutorial], page 121), select the 'File->Open' menu, and choose '`t1.geo`' in the input field. When using a terminal, you can also 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 module menu) and choose the required dimension in the context-dependent buttons ('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, choose 'Save' in the context-dependent buttons, or select the appropriate format with the 'File->Save As' menu. The default mesh file name is based on the name of the first input file on the command line (or '`untitled`' if there wasn't any input file given), 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 module menu, and follow the context-dependent buttons. 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 project file. You can edit this project file by hand at any time by pressing the 'Edit' button in the 'Geometry' menu and then reloading the project by pressing 'Reload'. For example, it is often faster to define variables and points directly in the project file, and then use the graphical user interface to define the lines, the surfaces and the volumes interactively.

Several files can be loaded simultaneously in Gmsh. The first one defines the project, while the others are appended ('merged') to this project. You can merge such files with the 'File->Merge' menu, or by directly specifying the names of the files on the command line. For example, to merge the post-processing views contained in the files '`view1.pos`' and '`view2.pos`' together with the geometry of the first tutorial '`t1.geo`', you can type the following command:

```
> gmsh t1.geo view1.pos view2.pos
```

In the Post-Processing module (select 'Post-Processing' in the module menu), two view buttons will appear, respectively labeled 'a scalar map' and 'a vector map'. A mouse click

[1] Nearly all the interactive commands have shortcuts: see Section 8.5 [Keyboard shortcuts], page 143, or select 'Help->Keyboard Shortcuts' in the menu.

on the 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. If you want the modifications made to one view to affect also all the other views, select the 'Apply next changes to all views' or 'Force same options for all views' option in the 'Tools->Options->Post-processing' menu.

Note that all the options specified interactively can also be directly specified in the ASCII input files. All available options, with their current values, can be saved into a file by selecting 'File->Save As->Gmsh options', or simply viewed by pressing the '?' button in the status bar. To save the current options as your default preferences for all future Gmsh sessions, use the 'Tools->Options->Save as defaults' button.

8.2 Non-interactive mode

Gmsh's second operating mode is the non-interactive (or 'batch') mode. In this mode, there is no graphical user interface, and all operations are performed without any user interaction [2]. For example, to mesh the first tutorial in non-interactive 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 8.3 [Command-line options], page 140.

8.3 Command-line options

Geometry options:

`-0` Parse all input files, output unrolled geometry, and exit.

`-tol float`
Set geometrical tolerance

Mesh options:

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

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

`-o file` Specify mesh output file name

`-format string`
Set output mesh format (msh, msh1, msh2, unv, vrml, stl, mesh, bdf, p3d, cgns, med)

`-bin` Use binary format when available

`-algo string`
Select mesh algorithm (iso, netgen, tetgen)

[2] If you compile Gmsh without the graphical user interface, i.e., with `./configure --disable-gui`, this is the only mode you have access to.

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

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

`-order int`
: Set the order of the generated elements (1, 2)

`-clscale float`
: Set characteristic length scaling factor

`-clmin float`
: Set minimum characteristic length

`-clmax float`
: Set maximum characteristic length

`-clcurv` Compute characteristic lengths from curvatures

`-rand float`
: Set random perturbation factor

`-bgm file` Load background mesh from file

`-constrain`
: Constrain background mesh with characteristic lengths

Post-processing options:

`-noview` Hide all views on startup

`-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:

`-nodb` Disable double buffering

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

`-theme string`
: Specify FLTK GUI theme

`-display string`
: Specify display

Other options:

`-` Parse input files, then exit

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

`-pid` Print pid on stdout

`-listen` Always listen to incoming connection requests.

`-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 this message

8.4 Mouse actions

In the following, for a 2 button mouse, *Middle button* = *Shift+Left button*. For a 1 button mouse, *Middle button* = *Shift+Left button* and *Right button* = *Alt+Left button*.

Move the mouse:

- Highlight the entity under the mouse pointer and display its properties in the status bar
- Resize a lasso zoom or a lasso selection/unselection started with *Ctrl+Left button*

Left button:

- Rotate
- Select an entity
- Accept a lasso zoom or a lasso selection started with *Ctrl+Left button*

Ctrl+Left button: Start a lasso zoom or a lasso selection/unselection

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 selection/unselection
- Pop-up menu on post-processing view button

Ctrl+Right button: Reset to default viewpoint

8.5 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

< Go back to previous context

> Go forward to next context

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

Ctrl+i Show statistics window

Ctrl+l Show message console

Ctrl+n Create new project file

Ctrl+o Open project file

Ctrl+q Quit

Ctrl+r Rename project file

Ctrl+s Save file

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-processing 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-processing views

Alt+s Hide/show geometry surfaces

Alt+t Loop through interval modes for visible post-processing 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

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: 2.0) that is independent of Gmsh's main version number.

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`) and post-processing datasets (`$NodeData`, `$ElementData`, `$ElememtNodeData`). 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).

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
$PhysicalNames
number-of-names
phyical-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 ...
...
$ElementEndNodeData
```

where

version-number
: is a real number equal to 2.0

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).

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 a mesh partition to which the element belongs. All tags must be postive integers, or zero. 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 152; for second order elements, the first order nodes are given first, followed by the nodes associated with the edges, followed by the nodes associated with the quadrangular faces (if any), followed by the nodes associated with the volume (if any). The ordering of these additional nodes follows the ordering of the edges and quadrangular faces given in Section 9.3 [Node ordering], page 152.

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.

number-of-real-tags
: gives the number of real number tags that follow.

number-of-integer-tags
: gives the number of integer tags that follow. By default the first *integer-tag* is interpreted as a time step number, the second as the number of field components of the data in the view, the third as the number of entities (nodes or elements) in the view, and the fourth as the partition index for the view data.

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.

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-elemenr and values are written in
  binary format ]
```

where

version-number
: is a real number equal to 2.0.

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), 3, 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 3 standard tags (the physical and elementary regions, and the mesh partition):

```
for(i = 0; i < number_of_triangles; i++){
  int data[7] = {num_i, physical, elementary, partition,
                 node_i_1, node_i_2, node_i_3};
  fwrite(data, sizeof(int), 7, file);
}
```

9.3 Node ordering

For all mesh and post-processing file formats, the reference elements are defined as follows. For high-order elements, the ordering of the edges/quad faces defines the ordering of high-order vertices associated with edges/faces.

```
Point:
        v
        |
        |
   -----1-----u
        |
        |

Line:
                  edge 1: nodes 1 2
        v
        |
        |
   --1-----2--u
        |
        |
```

```
Triangle:
                  edge 1: nodes 1 2
   v                   2:       2 3
   |                   3:       3 1
   |
   3
   |\
   | \
   |__\___u
   1   2
```

```
Quadrangle:
                  edge 1: nodes 1 2   quad. face 1: nodes 1 2 3 4
        v              2:       2 3
        |              3:       3 4
     4--|--3           4:       4 1
     |  |  |
   -----------u
     |  |  |
     1--|--2
        |
```

```
Tetrahedron:
                  edge 1: nodes 1 2
   v                   2:       2 3
   |                   3:       3 1
   |                   4:       4 1
   |                   5:       4 3
   3                   6:       4 2
   |\
   | \
   |__\2_____u
   1\ /
     \4
      \
       w
```

```
Hexahedron:
                  edge 1: nodes 1 2   quad. face 1: nodes 1 4 3 2
        v              2:       1 4              2:       1 2 6 5
        |              3:       1 5              3:       1 5 8 4
        |              4:       2 3              4:       2 3 7 6
   4----|--3           5:       2 6              5:       3 4 8 7
   |\   |  |\          6:       3 4              6:       5 6 7 8
   | 8-------7         7:       3 7
   | |   ----|---u     8:       4 8
   1-|---\-2 |         9:       5 6
    \|    \ \|        10:       5 8
     5-----\-6        11:       6 7
            \         12:       7 8
             w



Prism:
                  edge 1: nodes 1 2   quad. face 1: nodes 1 2 5 4
      v                2:       1 3              2:       1 4 6 3
    3 |                3:       1 4              3:       2 3 6 5
    |\|                4:       2 3
    | |                5:       2 5
    1_|2               6:       3 6
     \| 6              7:       4 5
      |_|_\___u        8:       4 6
       \|  \           9:       5 6
        4 __5
         \
          \
           w



Pyramid:
                  edge 1: nodes 1 2   quad. face 1: nodes 1 4 3 2
        v              2        1 4
        |              3        1 5
        |              4        2 3
    4---|---3          5        2 5
    | \ |  /|          6        3 4
    |  \ -/-|---u      7        3 5
    |  / 5\ |          8        4 5
    1/----\-2
           \
            \
             w
```

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 aplications.

9.4.1 MSH file format version 1.0

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 147. 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 152; for

second order elements, the first order nodes are given first, followed by the nodes associated with the edges, followed by the nodes associated with the quadrangular faces (if any), followed by the nodes associated with the volume (if any). The ordering of these additional nodes follows the ordering of the edges and quadrangular faces given in Section 9.3 [Node ordering], page 152.

9.4.2 POS ASCII file format

The POS ASCII file is Gmsh's old native pot-processing format, now superseded by the format described in Section 9.1 [MSH ASCII file format], page 147. 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 152. For second order elements, the first order nodes are given first, followed by the nodes associated with the edges, followed by the nodes associated with the quadrangular faces (if any), followed by the nodes associated with the volume (if any). The ordering of these additional nodes follows the ordering of the edges and quadrangular faces given in Section 9.3 [Node ordering], page 152.

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

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 157, 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.

10 Programming notes

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 autoconf. Practical notes on how to compile Gmsh's source code are included in the distribution. See Appendix B [Frequently asked questions], page 181, for more information.

10.1 Main code structure

Gmsh's code is structured in several libraries, roughly separated between the three main core modules (Geo, Mesh, Post) and associated utility libraries (Common, Numeric) on one hand, and graphics (Graphics) and interface (Fltk, Box) libraries 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).

10.2 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., add `-Wall` to `FLAGS` in the '`variables`' file);
2. always use the `Msg()` function to print information, errors, ...;
3. indent your files using '`utils/misc/indent.sh`';

10.3 Option handling

To add a new option in Gmsh:

1. create the option in the `Context_T` 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/GUI.cpp`';
5. optional: if no special callback is to be associated with the widget, add the handling routine `opt_XXX` to the OK callback for the corresponding option panel (in '`Fltk/Callbacks.cpp`').

11 Bugs, versions and credits

11.1 Bugs

If you think you have found a bug in Gmsh, you can report it by electronic mail to the Gmsh mailing list at gmsh@geuz.org. 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 8.3 [Command-line options], page 140 to see how to get this information).

See Appendix B [Frequently asked questions], page 181, and the 'TODO' file in the distribution to see which problems we already know about.

11.2 Versions

```
$Id: VERSIONS,v 1.406 2008-04-18 20:54:01 geuzaine Exp $

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 OpenCascade; 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: 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.

11.3 Credits

$Id: CREDITS,v 1.51 2008-03-21 09:39:43 geuzaine Exp $

Christophe Geuzaine
<cgeuzaine at ulg.ac.be>

and

Jean-Francois Remacle
<jean-francois.remacle at uclouvain.be>

Major code contributions to Gmsh have been provided by Nicolas Tardieu (help with the GSL and Netgen integration), Stephen Guzik (CGNS integration), Pascale Noyret (MED integration) and Jonathan Lambrechts (Fields).

Other code contributors include: David Colignon for new colormaps and lots of testing and bug reports; Patrick Dular for transfinite mesh bug fixes; Laurent Stainier for the eigenvalue solvers and for help with the MacOS port and the tensor display code; Pierre Badel for help with the GSL integration; Marc Ume for the original list code; Matt Gundry for the Plot3d mesh format; Jozef Vesely for help with the Tetgen integration; Koen Hillewaert for high order element mappings and other improvements.

The AVL tree code (DataStr/avl.*) and the YUV image code (Graphics/gl2yuv.*) are copyright (C) 1988-1993, 1995 The Regents of the University of California. Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of the University of California not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. The University of California makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

The trackball code (Common/Trackball.*) is copyright (C) 1993, 1994, Silicon Graphics, Inc. ALL RIGHTS RESERVED. Permission to use, copy, modify, and distribute this software for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both the copyright notice and this permission notice appear in supporting documentation, and that the name of Silicon Graphics, Inc. not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

The GIF and PPM routines (Graphics/gl2gif.cpp) are based on code copyright (C) 1989, 1991, Jef Poskanzer. Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation. This software is provided "as is" without express or implied warranty.

The colorbar widget (Fltk/Colorbar_Window.cpp) was inspired by code from the Vis5d program for visualizing five dimensional gridded data sets, copyright (C) 1990-1995, Bill Hibbard, Brian Paul, Dave Santek, and Andre Battaiola.

This version of Gmsh may contain code (in the contrib/ANN subdirectory) copyright (C) 1997-2005 University of Maryland and Sunil Arya and David Mount: check the configuration options.

This version of Gmsh may contain code (in the contrib/MathEval

```
subdirectory) based on GNU libmatheval, copyright (C) 1999, 2002, 2003
Free Software Foundation, Inc: check the configuration options.

This version of Gmsh may contain code (in the contrib/Netgen
subdirectory) copyright (C) 1994-2004, Joachim Sch"oberl: check the
configuration options.

This version of Gmsh may contain code (in the contrib/Tetgen
subdirectory) copyright (C) 2002, 2004, 2005, 2006 Hang Si: check the
configuration options.

This version of Gmsh may contain code (in the contrib/Metis
subdirectory) written by George Karypis (karypis at cs.umn.edu),
copyright (C) 1998, Regents of the University of Minnesota: check the
configuration options.

This version of Gmsh may contain code (in the
contrib/NativeFileChooser subdirectory), copyright (C) 2004 by Greg
Ercolano: check the configuration options.

Special thanks to Bill Spitzak, Michael Sweet, Matthias Melcher 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 OpenCascade 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, Eric Bechet, Laurent Champaney, Pascal Dupuis, 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, Ruth Sabariego,
Renaud Sizaire, Michel Benhamou, Emilie Marchandise.
```

Appendix A Tips and tricks

- Install the 'info' version of this user's guide! On your (Unix) system, this can be achieved by
  1. copying all '`gmsh.info*`' files to the place where your info files live (usually '`/usr/info`'), and
  2. issuing the command `install-info /usr/info/gmsh.info /usr/info/dir`.

  You will then be able to access the documentation with the command `info gmsh`. Note that particular sections ('nodes') can be accessed directly. For example, `info gmsh surfaces` or `info gmsh surf` will take you directly to Section 3.1.3 [Surfaces], page 43.
- Use emacs to edit your files, and load the C++ mode. This permits automatic syntax highlighting and easy indentation. Automatic loading of the C++ mode for '`.geo`' files can be achieved by adding the following command in your `.emacs` file: `(setq auto-mode-alist (append '(("\\.geo$" . c++-mode)) auto-mode-alist))`.
- Define common geometrical objects and options in separate files, reusable in all your problem definition structures.
- Save your preferred options with 'Tools->Options->Save as defaults'. To reset the default options, erase the `General.OptionsFileName` (usually '`.gmsh-options`' in your home directory) or use the 'Restore default options' button in 'Tools->Options->General->Output'.
- In the graphical user interface:
  - 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;
  - selecting the content of an input field, or lines in the message console ('Tools->Message Console'), copies the selected text to the clipboard;
- Read Appendix B [Frequently asked questions], page 181...

Appendix B Frequently asked questions

$Id: FAQ,v 1.83 2008-02-17 10:17:02 geuzaine Exp $

This is the Gmsh FAQ

****

Section 1: The basics

* 1.1 What is Gmsh?

Gmsh is an automatic three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. Its design goal is to provide a simple meshing tool for academic problems with parametric input and advanced visualization capabilities.

* 1.2 What are the terms and conditions of use?

Gmsh is distributed under the terms of the GNU General Public License. See the file 'doc/LICENSE' for more information, or go to the GNU foundation's web site at http://www.gnu.org.

* 1.3 What does 'Gmsh' mean?

Nothing ;-)

(Note that in the US, people tend to pronounce 'Gmsh' as 'Gee-mesh'. Yeehaa!)

* 1.4 Where can I find more information?

http://www.geuz.org/gmsh/ is the primary location to obtain information about Gmsh. You will for example find a complete reference manual as well as a searchable archive of the Gmsh mailing list (gmsh@geuz.org) on this webpage.

****

Section 2: Installation

* 2.1 Which OSes does Gmsh run on?

Gmsh is known to run on Windows 95/98/NT/2000/XP, Linux, Mac OS X, Compaq Tru64 Unix (aka OSF1, aka Digital Unix), Sun OS, IBM AIX, SGI IRIX, FreeBSD and HP-UX. It should compile on any Unix-like operating system, provided that you have access to a recent C and C++ compiler.

```
* 2.2 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.

* 2.3 What do I need to compile Gmsh from the sources?

You need a C and a C++ compiler (e.g. the GNU compilers gcc and g++)
as well as the GSL (version 1.2 or higher; freely available from
http://sources.redhat.com/gsl/) and FLTK (version 1.1.x, configured
with OpenGL support; freely available from http://www.fltk.org).

* 2.4 How do I compile Gmsh?

Just type './configure; make; make install'. If you change some
configuration options (type './configure --help' to get the list of
all available choices), don't forget to do 'make clean' before
rebuilding Gmsh.

********************************************************************

Section 3: General problems

* 3.1 Gmsh (from a binary distribution) complains about missing
libraries.

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
sources.

* 3.2 Gmsh keeps re-displaying its graphics when other windows
partially hide the graphical window.

Disable opaque move in your window manager.

* 3.3 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).

* 3.4 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).

* 3.5 How can I save GIF, JPEG, ..., images?

Just choose the appropriate format in 'File->Save As'.

* 3.6 How can I save MPEG, AVI, ..., animations?

See question 7.9.

********************************************************************

Section 4: Geometry module

* 4.1 Does Gmsh support NURBS curves/surfaces?

Yes, but only via STEP, IGES or BREP model import (not in .geo files).

* 4.2 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.).

* 4.3 How can I display only selected parts of my model?

Use 'Tool->Visibility'.

There are three main modes: 'Elementary entities', in which the
selections will apply to elementary geometrical entities; 'Physical
groups', in which the selections will apply to physical entities; and
'Mesh partitions', in which the selections will apply to mesh
partitions.
```

If the 'Recursive' option is set, selecting an entity also selects all its boundaries, recursively. For example, if 'Recursive' is set, selecting a surface will automatically select its boundary curves, as well as the boundaries of these curves (i.e., points).

In the 'Browser' tab, you can select which entities to show or hide in a list (several entities can be selected at once by dragging the mouse or by holding the Ctrl or Shift keys while clicking). In the 'Numerical input' tab, you can choose which entities to show or hide by explicitly specifying their identification tags. In the 'Interactive' tab, you can hide/show entities using the mouse.

4.4 When compiled with OpenCascade support, Gmsh crashes at startup

Try changing these environment variables, which govern how OpenCascade allocates memory:

```
export MMGT_OPT=0
export MMGT_MMAP=0
```

4.5 Can I edit STEP/IGES/BRep models?

Not yet. At the moment you can only change characteristic lengths and define physical groups. 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.).

****

Section 5: Mesh module

* 5.1 What should I do when the 2D unstructured algorithm fails?

Verify that the 1D mesh does not self-intersect. If it does, use a smaller characteristic length. If it doesn't, send us a bug report (including your geometry).

* 5.2 What should I do when the 3D unstructured algorithm fails?

Try the other 3D algorithms (Tool->Options->Mesh->General->3D algorithm). If none works, try to adapt the characteristic lengths in your input file so that the surface mesh better matches the geometrical details of the model. If nothing works, send us a bug report (including your geometry).

* 5.3 The quality of the elements generated by the 3D algorithm is very bad.

Use 'Optimize quality' in the mesh menu.

* 5.4 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.

* 5.5 Does Gmsh automatically couple unstructured tetrahedral meshes and structured hexahedral meshed using pyramids?

No. We need you help to implement this.

* 5.6 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.

* 5.7 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").

* 5.8 Does Gmsh support curved elements?

Yes, Gmsh can generate both 1st order and 2nd order elements. To generate second order elements, click on 'Second 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'.

* 5.9 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'.

* 5.10 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.

* 5.11 How can I display only the mesh associated with selected geometrical entities?

See question 4.3.

* 5.12 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. There is also one additional clipping plane available for "cutting" only the mesh (by keeping entire elements), in 'Tools->Options->Mesh->Cutting'.

* 5.13 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.

****

Section 6: Solver module

* 6.1 How do I integrate my own solver with Gmsh?

If you want to simply launch a program from within Gmsh, just edit the options to define your solver commands (e.g. Solver.Name0, Solver.Executable0, etc.), and set the ClientServer option to zero (e.g. Solver.ClientServer0 = 0).

If you want your solver to interact with Gmsh (for error messages, option definitions, post-processing, etc.), you will need to link your solver with the GmshClient routines and add the appropriate function calls inside your program. You will of course also need to define your solver commands in an option file, but this time you should set the ClientServer variable to 1 (e.g. Solver.ClientServer = 1). C, C++, Perl and Python solver examples are available in the source distribution in the 'utils/solvers' directory.

* 6.2 On Windows, Gmsh does not seem to find the solver executable. What's wrong?

The solver executable (for example, 'getdp.exe') has to be in your path. If it is not, simply go to the solver options (the second tab in the Solver dialog) and specify its location in the 'Executable' field.

* 6.3 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 question 6.1 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.

****

Section 7: Post-processing module

* 7.1 How do I compute a section of a plot?

Use 'Tools->Plugins->Cut Plane'.

* 7.2 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.

* 7.3 Can Gmsh generate isovolumes?

Yes, with the CutMap plugin (set the ExtractVolume option to -1 or 1 to extract the negative or positive levelset).

* 7.4 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.

* 7.5 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.

* 7.6 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.

```
* 7.7 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'.

* 7.8 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.

* 7.9 Is there a way to save animations?

Yes, using scripts. Have a look at 'tutorial/t8.geo' or
'demos/anim.script' for some examples.

* 7.10 Is there a way to visualize only certain components of
vector/tensor fields?

Yes, using 'Tools->Plugins->Extract'.

* 7.11 Can I do arithmetic operations on a view? Can I perform
operations involving different views?

Yes, with the Evaluate plugin.

* 7.12 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. A 3D version of this plugin is
  not available yet but it is on our TODO list.

- 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.

* 7.13 My code generates data "time step by time step", and thus cannot easily output Gmsh's multi-time-step post-processing files, where the values for all the time steps are given per element. How can I use Gmsh's post-processor in this situation?

Just create one view for each time step: Gmsh can handle an arbitrary number of views and it can deal with these separate views as efficiently as with a single multi-time-step view. The only disadvantage is that the total amount of disk space used is greater (since the node data is repeated for each time step).

In practice, depending on the size of the data set, you may want to store all the views in a single file or create one separate file for each view, which you can then load selectively (and thus reduce the memory required for the analysis). In any case you can use 'Tools->Options->Post-processing->View links' to apply options to multiple views at once, and the up and down arrow keys to loop through (animate) the views (instead of the left and right arrow keys for multi-time-step views).

Also note that if all the views are based on the same grid, Gmsh can combine the separate views into a multi-time-step view by using the 'View->Combine->Time Steps' menu, or by using the '-combine' command line option.

* 7.14 How can I see "inside" a complicated post-processing view?

See question 5.12.

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.

* 7.15 I am loading a valid 3D scalar view but Gmsh does not display anything!

In versions < 1.61, the default drawing mode for 3D scalar views was to draw iso-surfaces. If your data set was constant per element, Gmsh would not draw anything (a fix for this would be to run Plugin(Smooth), which would average the data on the nodes of the

```
grid)... This behavior has changed in version 1.61, and Gmsh now draws
the solution on the boundary of the elements by default. Iso-surfaces
are of course still available by setting 'Intervals type' to
'Iso-values' in 'Tools->Options->View->Range'.

Note that the most efficient way to visualize the dataset on the
boundary of the elements is to run Plugin(Skin) on the view: this will
extract the boundary of the dataset and only draw the data on this
boundary.
```

Appendix C License

Gmsh is provided under the terms of the GNU General Public License (GPL) with the following exception:

The copyright holders of Gmsh give you permission to combine Gmsh with code included in the standard release of Triangle (written by Jonathan Shewchuk), TetGen (written by Hang Si) and Netgen (written by Joachim Sch"oberl) under their respective licenses. You may copy and distribute such a system following the terms of the GNU GPL for Gmsh and the licenses of the other code concerned, provided that you include the source code of that other code when and as the GNU GPL requires distribution of source code.

Note that people who make modified versions of Gmsh are not obligated to grant this special exception for their modified versions; it is their choice whether to do so. The GNU General Public License gives permission to release a modified version without this exception; this exception also makes it possible to release a modified version which carries forward this exception.

End of exception.

GNU General Public License

Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

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

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, 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 show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

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

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in 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 any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.
9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.
10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH

HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

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

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

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

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

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

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place - Suite 330, Boston,
MA  02111-1307, USA.
```

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

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

```
Gnomovision version 69, Copyright (C) 19yy name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type
'show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type 'show c' for details.
```

The hypothetical commands '`show w`' and '`show c`' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than '`show w`' and '`show c`'; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
'Gnomovision' (which makes passes at compilers) written by James Hacker.
```

signature of Ty Coon, `1 April 1989`
`Ty Coon, President of Vice`

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Concept index

L

M

N

O

P

Q

R

S

T

U

V

W

Syntax index

A

B

C

D

E

F

G

H

I

L

M

R

S

T

V