MEDIT : An interactive Mesh visualization Software

Pascal Frey

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

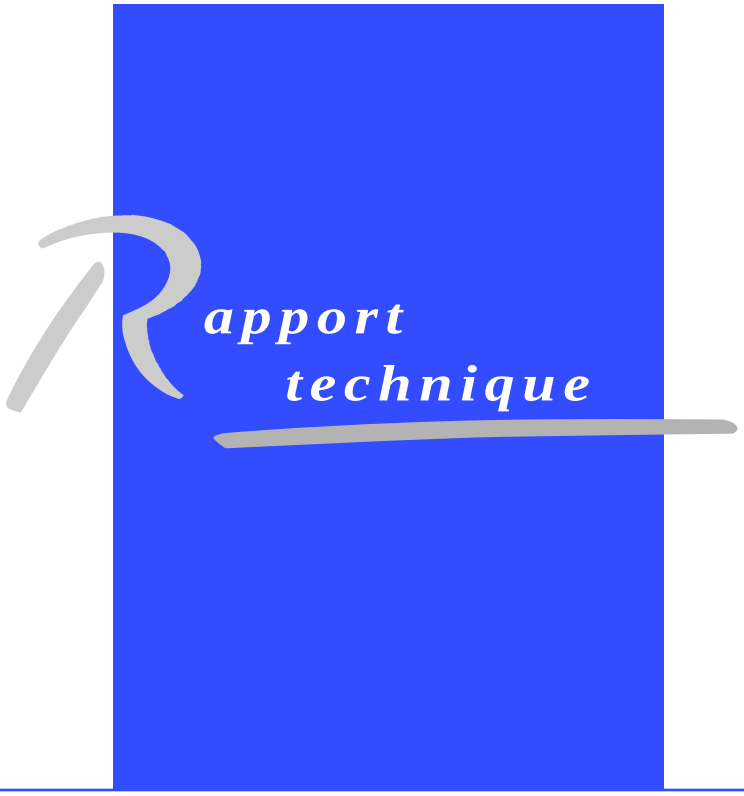
MEDIT
An interactive mesh visualization software

Pascal J. FREY

No 0253

December 3, 2001

THÈME 4



ISSN 0249-0803

MEDIT
An interactive mesh visualization software

Pascal J. FREY

Thème 4 — Simulation et optimisation
de systèmes complexes
Projet Gamma

Rapport technique n°0253 — December 3, 2001 —41 pages

Abstract:
This technical report describes the main features of MEDIT*, an interactive mesh visualization tool developped in the Gamma project at INRIA-Rocquencourt. Based on the graphic standard OpenGL, this software has been specifically designed to fulfill most of the common requirements of engineers and numericians, in the context of numerical simulations. This program is rather intuitive and, therefore, the user does not really need any specific learning stage prior to be able to play with it.

In this document, the user will learn how to visualize and manipulate mesh data structures via the mouse, how to deal with scalar/tensor values associated with a mesh and how to deal with more complex features (e.g., animations, postscript or image output creation, etc.).

This document describes the features of the current version : release V2.1 (June, 2001) and, as such, replaces the previous document [3].

Key-words: visualization, post-processing, mesh

(Résumé : tsvp)

* This software wa registered with the APP under n° IDDN.FR.001.410023.00.R.P. 2001.000.10800 on january 25, 2001.

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
Téléphone : 01 39 63 55 11 - International : +33 1 39 63 55 11
Télécopie : (33) 01 39 63 53 30 - International : +33 1 39 63 53 30

MEDIT

Résumé :
Ce rapport technique décrit les principales fonctionnalités de MEDIT[†], un logiciel interactif de visualisation de maillages développé au sein du projet Gamma à l'INRIA-Rocquencourt. Basé sur le standard graphique OpenGL, ce logiciel a été spécialement conçu pour répondre aux besoins les plus courants des ingénieurs et numériciens, dans le contexte de simulations numériques. Le fonctionnement de ce programme est relativement intuitif.

Dans ce rapport, on décrit comment visualiser et manipuler des structures de données de maillages à l'aide de la souris, comment représenter des champs scalaires/tensoriels associés à des maillages et comment utiliser des fonctions complexes (animations, sorties postscript, etc.).

Ce document décrit les fonctions de la version courante : version V2.1 (juin, 2001) et, à ce titre, remplace le précédent rapport [3].

Mots-clé : visualisation, post-traitement, maillage

[†] Ce logiciel a été enregistré à l'APP sous le numéro IDDN.FR.001.410023.00.R.P. 2001.000.10800 le 25 janvier 2001.

Contents

1 Scientific visualization

1.1 Context

Numerous applications of numerical simulations based on finite element/volume methods involve a mesh of the computational domain as spatial support. In this context, it is often necessary to visualize this mesh and the solutions associated with mesh entities. This is typically the aim of a scientific visualization tool (also called a post-processing tool). Hence, the use of graphical devices allows, provided the software is well-suited to the problem considered, the user to appreciate and even to analyze a mesh and/or a solution. To be efficient and pertinent, this tool must be fast and should offer various features, if possible, in a user-friendly environment (*i.e.*, convivial and interactive).

Among the various graphical software already available, it would seem reasonable to identify one that matches the previous requirements. Nevertheless, a preliminary study has convinced us that almost 90% of the public concerned (engineers, students, researchers, etc.) is only concerned with roughly 10% of the potential capabilities of such tools. But more meaningfull, these tools are often considered as inappropriate (*i.e.,* the right feature is always missing !) or too difficult to handle (*i.e.,* too many parameters to adjust before getting to the right view).

This situation has led us to develop a tool, deliberately limited in its functionnalities, aimed at *numericians* more concerned about easily locating critical regions in a dense mesh than focussed on graphical features (ray tracing, for instance). More importantly, the functionnalities of this tool should reflect the concerns and remarks of the users rather than the *virtuosity* of the software developers. This been stated, `MEDIT` cannot be considered as a substitute to the existing graphical tools, but is more a *cheap* alternative to users that need to quickly look at a result without messing around with a complex GUI. As you may understand, this approach is rather iconoclastic in a time where virtual reality and GUI are the lifeblood of computer graphic.

1.2 Medit overview

`MEDIT` has been designed as a user-friendly mesh visualization tool (2D, 3D and surfaces). It allows to load and process large meshes relatively fast on a workstation with a graphical accelerator (approximately 3 millions elements per second on a HPUX 9000 workstation with a FX10 graphic card). Another concern was to write a portable code that runs on a wide variety of platforms with the same look-and-feel. Therefore, the choice of the graphic Application Program Interface `OpenGL` and the associated library `GLUT` was almost seen as natural. Since its introduction in 1992, `OpenGL` has become the industry's most widely used and supported 2D and 3D graphics API.

Features. `MEDIT` offers various post-processing features for the visualization of meshes and associated numerical data (scalar/tensor fields). The mesh data structure can be easily and interactively manipulated using the mouse and all transformations (rotations, translations, etc.) are defined in this very intuitive manner. The user can pick a mesh entity and, for instance, may hide the such-defined sub-domain, thus 'peeling' the mesh structure like an orange (this is usefull for seeing the interior of a complex model). If the mesh structure has been throroughly described via the `mesh` format, specifed mesh entities can be highlighted (ridges, corners, normals, etc.). If a solution has been computed, a scalar or tensor field can be easily associated with mesh vertices (or elements), and then `MEDIT` offers various ways of rendering this kind of data (iso-lines, iso-surfaces, color map, streamlines, etc.).

Output files (several postscript or image formats supported) can also be created to save a specific view that can be later imported in a text processing software.

MEDIT allows also to easily create animations (e.g. animated GIFs) from a set of mesh files or by animating a single mesh. Another commonly used feature concerns the possibility of linking several views together, each of which corresponding to a mesh structure.

For all these reasons, MEDIT can be considered as a valuable auxiliary for mesh debugging as well as a powerfull mesh visualization tool. Its efficiency is partly due to the graphic kernel it uses and also to the context it has been developed for.

Remark 1.1 *From a practical point of view, this document is divided into 7 sections. However, the main concepts and features of* MEDIT *are described in sections 2 to 4, to which the user is referred. Section 4, especially, has been written in such a way that the user should easily understand how the software works and how to control the various options. Section 5 describes the code features in an exhaustive manner.*

2 Input data

The mesh geometry is described in a comprehensible file format. Solutions and related information are described in separate files associated with the mesh file.

2.1 Geometry description

The mesh geometry is described preferably using the `mesh` file format (see Appendix, Section 7.2, for a complete description of this format). For the sake of compatibility, the "old" `msh2` data format can also be used in this release. In addition to these formats, this version of `MEDIT` now supports a new `gis` format, specially designed for terrain description.

- The `mesh` format.
  This format is composed of a single (ASCII or binary) file, `xxx.mesh` or `xxx.meshb`. This file contains all the information needed to describe entirely the surface mesh and the underlying geometry. It is organized as a series of fields, identified by keywords. In addition to the vertex coordinates and the list of faces and elements, it allows one to specify additional information such as corners, edges, ridges, required entities, etc.

- The `msh2` format.
  This very crude format, intended for surface mesh description only, is composed of two ASCII files, `xxx.points` and `xxx.faces` describing the geometry (vertex coordinates) and the topology (list of elements) of a surface mesh, respectively.

- The `gis` format.
  This simple format is especially designed for cartesian surfaces or terrains and is composed of a single (ASCII or binary) file `xxx.gis`. The file structure consists in a short sequence indicating the resolution and the spatial location of the surface, followed by a series of scalar values representing the height values at the nodes of a regular grid.

2.2 Related information

Numerical (scalar or tensor) values can be associated with the mesh nodes or elements. They are described as a single data file, `xxx.bb`. By default, after reading the input file (`xxx.mesh`, `MEDIT` will also look for a `xxx.bb` file. Depending on the nature of these data, a color map or streamlines or iso-surfaces can be visualized in the graphic window by selecting the appropriate option in the main menu. A `xxx.iso` file can also be supplied that contains information to draw streamlines or streamribbons (this file will be read only if the data correspond to tensor values).

2.3 Configuration file

A special `.medit` file allows the user to set up several parameters used in the code (window size, color background, etc.). The file structure is organized as a series of fields, identified by keywords and a list of scalar values.

3 Technical information

3.1 Language, platforms

The program is entirely written is Ansi C. The current version consists of approximatively $15,000$ lines of optimized code. This code is highly portable and has been successfully compiled and tested on all major computer architextures (*i.e,* HP, SUN, SGI, IBM, Intel-based PC, etc.) and operating systems (Unix/Linux, WindowsNT, Max OS, etc.).

3.2 Memory requirement

The memory is dynamically allocated during the data reading stage. On a classical workstation (512 MegaBytes of RAM), the user can load and visualize large data files (several millions elements). If the local memory is not sufficient to store the mesh structure, the following message is issued :
`ERR 1000, proc, UNABLE TO ALLOCATE MEMORY.`
For instance, with the given demo file (see Section 4.2), the program requires roughly 12 Megabytes of memory to store all data structures (mesh + graphic lists). Similarly, a surface mesh having more than a million vertices requires about 22 MegaBytes.

Notice that part of the memory can be allocated on a graphic card, if any. For some specific options (e.g. streamlines), additional memory may be needed and allocated at the time the option is selected.

3.3 Performances

`MEDIT` performances are strongly dependent on the CPU (processor) type and the graphic card used (e.g. wireframe rendering speed range from $20,000$ poly/sec on an old 200 Mhz P3 without graphic accelerator to more than 3 millions poly/sec for a HP9000 with a Fx10 card). Notice however, that the user will have access to the same features, irrespective of its workstation configuration.

3.4 Distribution

An evaluation version of `MEDIT` (including all options and not limited in time) is available to download freely from the web site :
`http://www-rocq.inria.fr/gamma/medit/medit.html`
This version has been compiled on Linux PC architecture and was build using MESA (a public domain alternative to `OpenGL`). Therefore, it should run slower on your machine because it may not take advantage of the graphic accelerators. However, this version allows the user to understand and evaluate the main principles and features of the code.
For any question regarding the software, please contact the author :

`Pascal J. FREY`
`INRIA, Domaine de Voluceau`
`BP 105, 78153 Le Chesnay cedex, France`
`Email: pascal.frey@inria.fr`
`http://www-rocq.inria.fr/gamma/medit/medit.html`

4 Medit at a glance

This section presents the various features offered by MEDIT. The user is advised to read this section in front of his computer, in order to observe and experiment the behavior of MEDIT in an interactive way. The first focus is put on the basic commands of the software, then the various features and options are discussed in more details.

4.1 Installation

This section assumes you have downloaded (or purchased) a binary version of MEDIT, hereafter refered to as medit. You have to also download the file meshes.tar.gz which contains several mesh examples (disk space requirement : xx Mb).

1. First, copy the executable to a directory where you want to run the code (check the write permissions).

2. Then, extract the mesh examples with the command :
   `tar xzvf meshes.tar.gz`
   or `gunzip meshes.tar.gz ; tar xvf meshes.tar`
   This will create the directory INRIA.dir that contains meshes and related files.

After this preliminary stage, you can check if MEDIT has been correctly installed by typing :
`medit -i`
in a shell window. You should get a result similar to (here on a HPUX machine) :

```
Graphic info:
 GL Vendor:      Hewlett-Packard Company
 GL Version:     1.1 Revision 1.20
 GL Renderer:    lib35acda30

  RGBA Mode
  Double Buffer
  Index Bits            24
  RGBA Bits              8  8  8 0
  Accum RGBA Bits       16 16 16 0
  Depth Bits            24
  Stencil Bits           4
```

If you are unable to get such result, check the variables (on Unix/Linux systems only) :

1. PATH that should contain the directory where medit has been copied and

2. LD_LIBRARY_PATH that indicates the directory containing the graphic libraries.

If the problem persists, contact the author (see Section 3.4).

To use MEDIT you need to have a 3-buttons mouse. Also, a graphic card (3D accelerator) is highly recommended for large meshes visualization ($> 100,000$ elements).

4.2 Getting started

Once the distribution has been installed and properly configured to your system architecture, MEDIT is ready to be used. In order to start MEDIT, change to the directory where the examples have been installed and simply type in the following command in an active window :
`medit part.mesh`
which launches the graphic software and reads the data file `part` (in `mesh` format). If the name of the file does not contain the `.mesh` extension, the program will look for files in :

1. `mesh` format (binary then ascii),
2. `msh2` format (crude file structure) and then,
3. `.gis` format (terrains), respectively.

Notice that the binary format `.meshb` has always priority over the corresponding Ascii format (for the sake of efficiency). If the command line does not contain any file name, then the software will prompt for one.

Remark 4.1 *In addition to a file describing a mesh topology,* MEDIT *also searches for a* `xxx.bb` *file which would possibly contain a related solution.*

After a short while (depending on the size of the mesh), the mesh is loaded and the following message is displayed in the shell window :

```
  -- Medit,  Release 2.1 (April, 2001)
     Copyright (C) INRIA, 1999-2001.

 Loading data file(s)
  Reading part.mesh
    Vertices      28694  Corners  50
    Triangles     14890
    Tetrahedra   150779
    Edges          1905  Ridges   1905
    Bounding box:  x:[-0.05  0.05]  y:[-0.033333  0.033333]  z:[-0.033...
  Reading part.bb
    Solutions     28694
  Input seconds:     2.79

 Building scene(s)
  Creating scene 1
   Reading DEFAULT.medit
   Identifying sub-domains
   Sorting 32 materials
   Computing 3D scene
  Scene seconds:     0.05

 Rendering scene(s)
```

At the same time, a graphic window should pop up, within which the mesh is displayed in hidden line mode (Figure 1, left-hand side). As there is no GUI attached with MEDIT, no other panel window will appear, as the user may already have noticed. All commands and features can be activated via key-strokes or the mouse. Now, move the cursor of the mouse in the main graphic window. Moving the mouse while keeping the left button pressed will rotate the mesh, in the direction of the mouse displacement. Releasing the button freeze the mesh position. Similarly, the middle button allows the user to translate the mesh.

Remark 4.2 *If the mesh disappears from the screen, simply type* `'i'` *on the keyboard to restore its initial position.*

A pop-up menu can be activated by pressing the right button of the mouse. It allows the user to select various options. For instance, place the mouse cursor on the *RenderMode* line and select the option *Shading+lines* from the relative sub-menu. The mesh will now look like Figure 1 (right-hand side). Try selecting other modes to see the various ways of rendering a mesh.

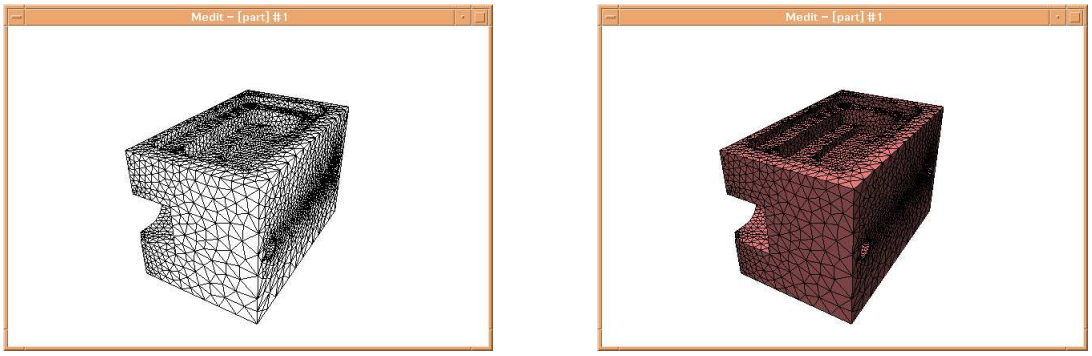


Figure 1: *Main graphic window created for rendering the mesh.*

MEDIT offers a wide variety of keyboard shortcuts to access most of the possible options. For instance, typing `'l'` toggles the mesh edges, `'c'` toggles the color. Other useful options include `'f'` to toggle mesh faces, `'b'` to change the background color, `'A'` to toggle the axis, `'g'` to display the geometric items (corners, ridges, etc.). The option `'z'` and `'Z'` allows to zoom-in or zoom-out the mesh. Typing `'h'` will display the list of all available keyboard shortcuts in the shell window.

Remark 4.3 *Notice that* MEDIT *is* **case sensitive**. *Hence,* `'a'` *does not produce the same result as* `'A'`.

4.3 3D meshes

As MEDIT can be used for debugging purposes, it is sometimes desirable to define a cutting section through a tetrahedral or hexahedral mesh. This can be done by selecting the option *[F1] Toggle clip* of the menu *Clipping*. This results in the activation and the display of the cutting plane $x = x_G$, x_G being the x-ccordinate of the barycenter of the mesh vertices (Figure 2, left-hand side). The current plane equation is written at the bottom of the graphic window. The plane section can be modified interactively by pressing the key function `'F2'`. This indicates that the selected object is now the plane, hence when pressing the left (resp. middle) button of the mouse a rotation (resp. translation) is applied directly on the plane (Figure 2, right-hand side). Notice that in this example, all tetrahedral elements intersected by the plane are displayed ('porcupine' effect). However, it is possible to get a 'clean' planar cut section by turning on the option *Toggle capping* in the *Clipping* menu.

Remark 4.4 *For efficiency purposes, it is advised to store the boundary elements of the volume (*i.e., *the surface triangles) in the input file. In this case, only the boundary elements will be displayed if no cutting plane is activated.*

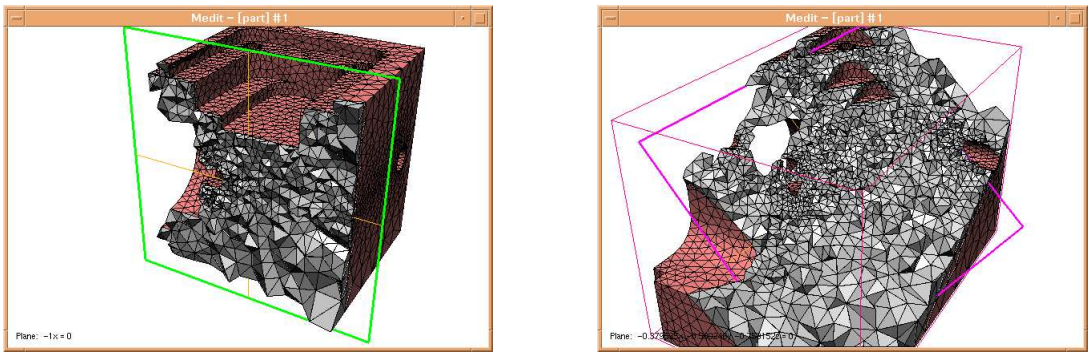


Figure 2: *Cutting section through a tetrahedral mesh (left-hand side) and after rotating the plane and zooming the scene (right-hand side).*

4.4 Post-processing options

`MEDIT` offers a few post-processing options, for instance the possibility of visualizing a scalar, vector or tensor field associated with the mesh file using a color map. In this case, a file `.bb` must be associated with the mesh input file. In our example, the file `part.bb` has been loaded with the `part.mesh` file; it contains scalar values associated with the mesh nodes. To display the scalar values using a pseudo-color map, simply type `'m'` (Figure 3, left-hand side). Instead of displaying a color map, `MEDIT` also offers the option of looking at the iso-lines, *i.e.*, the lines of iso-values traced onto the surface (Figure 3, right-hand side).

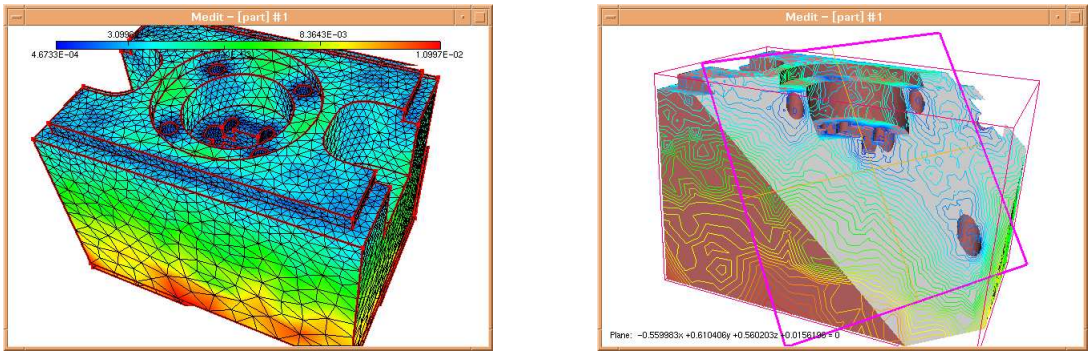


Figure 3: *Post-processing capabilities : visualization of a scalar field associated with mesh vertices. Left : scalar map on the surface. Right : iso-lines through a cutting plane.*

In case of a vector field, the user will have the possibility of tracing streamlines. These lines are constructed from starting points in space, the points being defined in a `.iso` file associated with the data.

4.5 Output options

`MEDIT` offers three options for saving bitmap postscript files of the screen : color, grey and b/w. Notice that the size (in cm) as well as the density (in dot per inch) of the postscript image can be controlled (int the control file, see next Section). Hence, the size of the poscript file is neither related to the size of the graphic window nor to the physical size of the graphic device. To this end, chose for instance option *Hardcopy EPS (color)* from the menu *File*. This feature allows the user to insert figures in a text file, for instance a LaTeX document.

4.6 Picking and selecting

The user can pick a mesh element (triangle, quadrilateral, tetrahedron or hexahedron) using the mouse. To this end, simply place the mouse cursor on an element and click on the mouse left button while pressing the `Shift` key. Selected items appear in different color on the screen. At the same time, information about this element are written in the shell window, like :

```
Picking result :
 Triangle  4616 : 3090, 4925, 5473    ref : 508 [MAT12]
   normal : 0.000000 0.000000 -1.000000
 vertex    3090 : -0.026549 0.010483 -0.033333   ref 0
 vertex    4925 : -0.029384 0.002740 -0.033333   ref 0
 vertex    5473 : -0.034578 0.009045 -0.033333   ref 0
```

Conversely, the user may want to locate a specific item on the screen (e.g. *where is triangle 1000 ?*). To this end, after typing '#' in the graphic window the user will be prompted to enter the number of the item to be visualized in the shell window. All items having the same number (*i.e.*, vertex 1000, triangle 1000, tetrahedron 1000) will be displayed in different color in the graphic window, with the vertex numbers.

Typing `'V'` when an item has been selected, allows the user to change the center of rotation to the barycenter of the designated item.

4.7 Shrink

On a very complex 3D geometry, it may be usefull to slightly disconnect the elements to each other in order to get a better understanding of the mesh topology. This feature correspond to the shrink option : type `F5` in the graphic window will toggle this feature. The key `F6` (resp. `F7`) will increase (resp. decrease) the shrink value (see Figure 4, left-hand side).

4.8 Terrain visualization

`MEDIT` also offers the possibility of visualizing terrains or cartesian surfaces using a 'fligh-simulator' mode. This mode is automatically activated when reading a file with the extension `.gis`. However, at any time, the user can switch between the usual display mode and this mode by typing `'J'`. Turning on the animation mode (using key `'a'`) will allow the user to fly through the terrain at constant speed, using the mouse to control the control the trajectory. The speed can be adjusted using keys `'+'` and `'-'`. When this mode is activated, a control panel shows up to indicate the current elevation and azimuth.

To get an overview of this feature, the user is advised to load the file `terrain.gis` from the distribution archive (in the `INRIA.dir` directory and to play with it.

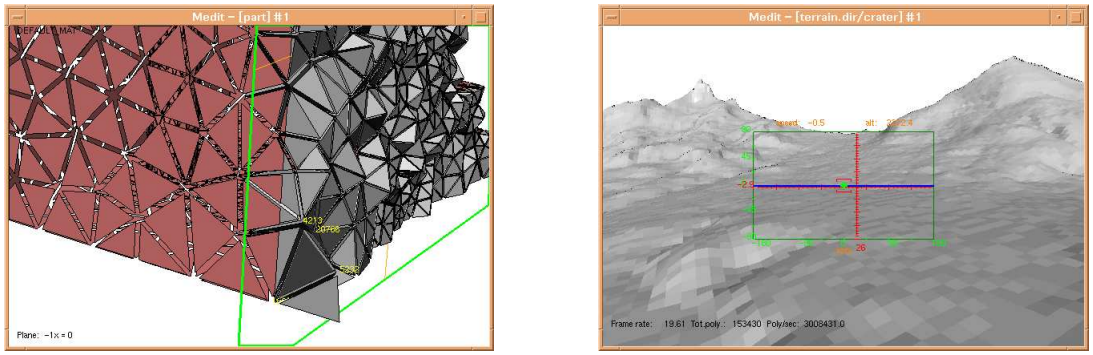


Figure 4: *Left : example of picking an item and shrinking mesh elements. Right : example of interactive visualization of terrain.*

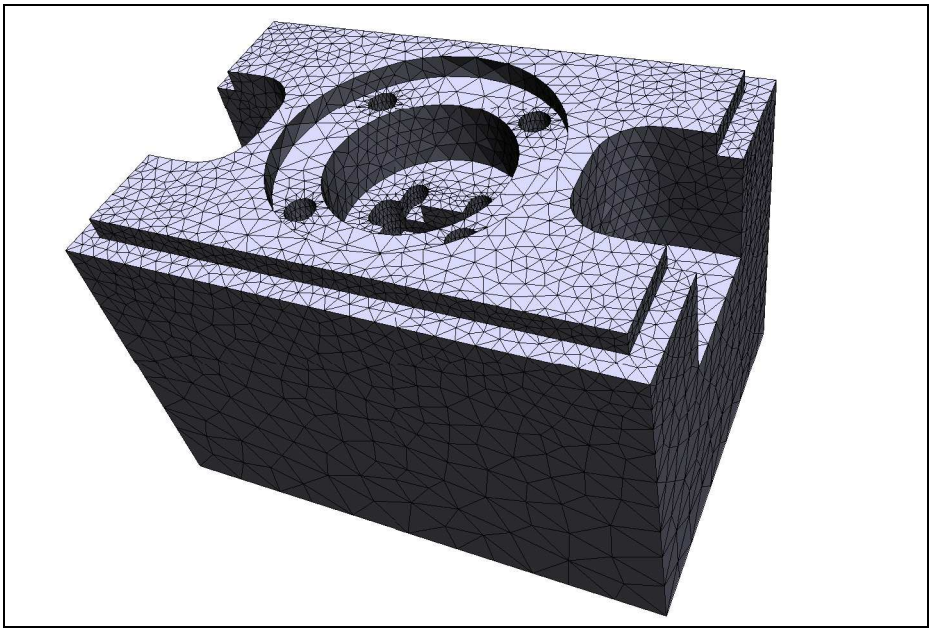


Figure 5: *Example of the output option : creation of a color postscript bitmap (12cm, 300dpi).*

5 How to use MEDIT

This section will describe in details the various options and features of the graphic software MEDIT. As it is claimed to be an interactive tool, the user is advised to sit in front of its computer and to play with the software in order to understand the concept of MEDIT. Remember that there is no harm in fooling around with it as the data are not modified by the software (original meshes are preserved). The following paragraphs will explain what the user can expect from this software and the better way of getting it.

5.1 Command line and options

MEDIT is usually invoked via a single line command. At this level, only a few options and parameters can be specified.

5.1.1 Command line

The usual way of launching MEDIT is to enter the following command in a shell window : `medit InputFile`
If no input file name is supplied, the code will ask for one. Notice that the command `medit -h` prints the list of the available options and parameters on the shell window :

```
Usage: medit [options] [f1 .. fn]

** Generic options :
-d       Debug mode (lot of info)
-fs      Fullscreen mode
-h       Print this message
-i       Print info on OpenGL configuration
-v       Turn off quiet mode
f1..fn   Input data file(s)

** Graphic options
-a start stop    Animation sequence
```

Notice that multiple input file names lead to open several graphic windows (*i.e.,* a mesh is associated with a window).

5.1.2 Options and parameters

Usually, an option corresponds to a specific data processing. By default, no option is required. The option `-a start stop` indicates that MEDIT is used to visualize a series of meshes : `xxx.start.mesh` to `xxx.stop.mesh`, where `start` and `stop` represent the range of the animation (see Section 5.5.1).

In addition, to the options, a few parameters can be specified in a separate `.medit` file associated with a mesh file. By default, MEDIT searches for a `xxx.medit` file (for file `xxx.mesh`) or `DEFAULT.medit`. This file is organized as a series of fields, each composed of a keyword and related scalar values (see Section 6 for a complete description of this structure).

5.2 Interaction

This small section explains how to load a mesh and interact with the graphic object associated with this mesh.

5.2.1 Loading files

To load a mesh data file, enter the name of this file directly on the command line (see Section 5.1.1). If no file name is supplied, `MEDIT` prompts the user for one :

```
File name(s) missing.  Please enter :
```

The user should then enter a valid file name. If no extension is provided, `MEDIT` assumes that a `mesh` file is to be read, first in binary, then in Ascii format. If no such file is found, the code will then look successively for a `msh2` file and a `gis` file.

`MEDIT` is primarily intended for visualizing mesh elements. However, a data file containing only vertices is considered as valid (use option `'g'` to see the vertices). The list of error messages related to data reading is given in Appendix.

If a `.bb` file exists (having the same basename as the mesh file), it will be loaded at the same time. In this case, the number of data in this file must be compatible with the mesh elements or mesh vertices to be considered as related data. Otherwise, these data will be automatically discarded and an error message issued.

As already indicated, `MEDIT` will also look for a `.medit` file that is used to change the values of rendering parameters. If no such file exists (nor the `DEFAULT.medit`), default values are assigned to rendering parameters (window size, background color, material colors, etc.).

Once a mesh file has been successfully loaded, information related to the mesh entities are displayed in the shell window. Then, a graphic window is opened in which the current mesh is displayed (by default with hidden lines removal). Notice that if multiple file names have been supplied in the command line, multiple graphic windows will be opened. This option is especially usefull to compare two meshes of the same domain, for instance before and after a specific treatment (as different views may be linked together, see Section 5.4.2).

Remark 5.1 *As there is no GUI, the user must quit* `MEDIT` *and rerun it each time a new mesh has to be loaded.*

Remark 5.2 *The file format determines automatically the viewing mode : 'perspective' for classical meshes (formats* `mesh` *and* `msh2`*) and 'flight' for terrains (format* `gis`*).*

5.2.2 Interacting with the mesh

After reading a mesh file, the mesh data structure is converted into a more suitable, graphic, data structure. A display list (*i.e.*, a list of graphic primitives) is built in order to speed up the rendering process. This stage is usually not very costly (only a few seconds for a very large mesh on a workstation). It is identified in the shell window by the following lines :

```
 Building scene(s)
  Creating scene 1
```

The converted mesh structure is then displayed in a graphic window. The user can directly interact with this graphic object via the mouse. Moving the mouse while pressing the left button rotates the mesh in the direction of the displacement. To stop the movement, simply release the mouse button. To translate the mesh, do the same with the mouse middle button (you can also use $\leftarrow$, $\rightarrow$, $uparrow$ and $downarrow$). To reset the mesh to its original position (centered on the screen), type `'i'`. When pressing the right button, a menu pops up and offers various choices that will be discussed hereafter.

If the mesh structure is very large, rotating it may severely impact the performances (*i.e.*, the display cannot be refreshed in real time). To overcome this problem, the user can select the interactive (or bounding box) mode, by typing `'I'`. If enabled, only the bounding box of the domain is displayed when manipulating the mesh using the mouse. The mesh will be displayed again as soon as the mouse button is released.

Automation. To automatically rotate a mesh on the screen, simply type `'a'` to toggle the *automation mode*. After entering this mode, any impulse via the mouse will define a rotation axis and a rotation angle that will be applied on the mesh automatically. The rotation angle is defined by the amplitude of the mouse displacement. To stop (or change the rotation), type `'a'` or move the mouse. Translations can also be applied (use the mouse middle button to initialize the transformation).

Remark 5.3 *For the sake of efficiency, when both surface and 3D elements are provided, the surface elements are supposed to correspond to the boundary of the domain and will be displayed only. In such a case, the only way of looking at the 3D elements is to activate a cutting plane through the volume.*

Zooming and scaling. Apart rotations and translations, the user can also zoom in and out the scene. This feature corresponds to decreasing (resp. increasing) the field of view angle and can be controlled by typing `'z'` (resp. `'Z'`). This feature may result in a large deformation of the mesh structure (due to the projection). An alternative to this option, is to use `'+'` (resp. `'-'`) in order to get closer (resp. farther) from the mesh.

Sometimes, it may be more convenient to delineate a small region on the screen and to resize this small portion to the whole screen size. To this end, press the `ctrl` key while moving the mouse with the left button. This will define a dynamic rubberband. The portion of screen delimited by this rectangle will be mapped onto the screen when typing `'z'`. To zoom out a specific region, proceed as above but using `'Z'`.

5.3 Classical features

This section describes the various possibilities of `MEDIT` from a practical point of view, *i.e.*, following the logical options of the main menu. This document does not claim for exhaustivity, as several features are context-dependent, however, the user should find here all ingredients to be able to use `MEDIT` in a profitable way.

5.3.1 Rendering options

Basically, these options concern the way the mesh is rendered on the screen. The user expects to see, and possibly understand, the mesh topology (the node connections) and the mesh geometry (the overall domain shape). Therefore, a mesh viewer should offer various features to show or hide specific mesh entities, to display entities numbers, and to focuss on small areas of interest.

The second line of the main menu, *Render mode*, concerns the basic rendering options. Selecting an option consists in showing or hiding a specific type of mesh entity (faces or edges) or mesh attribute (color). In practice, a binary flag (on/off) is associated with each option and can be modified at any time. The effect of switching a flag will be seen on the next redraw of the current scene on the screen. By default, the option *Hidden lines* is selected, meaning that both the face and the edge flags are turned on, while the color flag is turned off. Hence, all polygon edges are drawn in the color opposite to the background (each color component complemented to 1), and the polygon interiors are filled with the background color. The following table summarizes the various possible choices.

OPTION	RENDERING MODE	Shortcuts
Wireframe	shows polygons (polyhedra) outline	`'l'`
Depth lines	colors polygons (polyhedra) outline	`'c' 'l'`
Hidden lines	hidden faces removed (face color = background)	`'f' 'l'`
Shading	shaded (colored) polygons (polyhedra)	`'c' 'f'`
Shading+lines	shaded polygons (polyhedra) + outline	`'c' 'f' 'l'`

Table 1: *Possible choices for rendering modes and associated keyboard shortcuts.*

All proposed options in the menu correspond to pre-defined combinations of attributes and entities, thus avoiding multiple selections in a menu and making any change more efficient. Actually, all possible options (flags) can also be selected (toggled) using keyboard shortcuts. For instance, the edge and facet flags are (de)activated by typing `'l'` and `'f'`, respectively and the color flag is switched when typing `'c'`.

As the user may notice, all polygons are displayed using the same default color, irrespective of the reference numbers attached to the facets. If the data contains multiple facets references, typing `'e'` turns the color material flag on. In this mode, each facet is colored with a color depending on the associated reference. This is especially useful for visualizing different sub-domains in different colors. In Section 5.4.5, we will see of to configure these colors to achieve more elaborate (realistic) rendering.

As pointed out in the previous section, it is sometimes desirable to shrink the facets (elements) slightly around their barycenter. This rendering mode is mainly aimed at getting a better understanding of the mesh topology. The key `F5` toggles the corresponding option, key `F6` (resp. `F7`) increases (resp. decreases) the amount of shrink between the elements. The shrink mode can be applied to both 2D (planar and surface) and 3D elements.

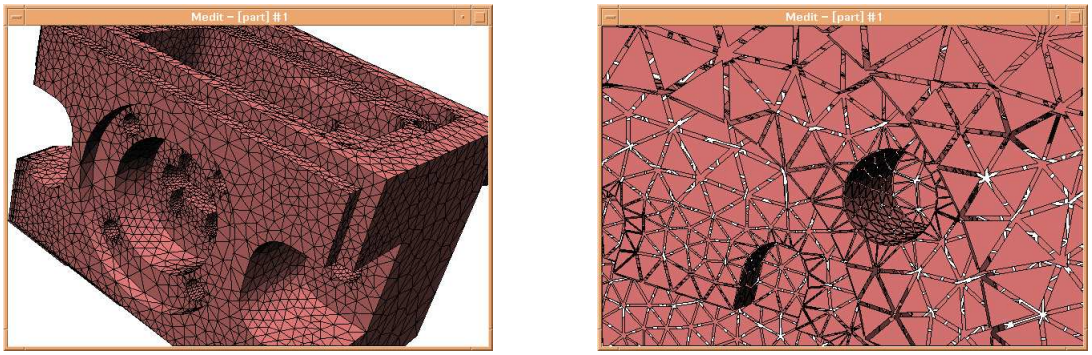


Figure 6: *Examples of rendering option :* Shading+lines *selection (left-hand side) and* Shrink *flag activated (right-hand side).*

5.3.2 Displaying specific items

Several predefined items can be associated with a mesh and visualized on the graphic display. These extra entities ar accessible via the *Items* menu. Hence, axis can be made made visible by typing `'a'`, the bounding box by typing `'B'`, an underlying grid by typing `'G'`. Similarly, it is possible to display vertex or face numbers by typing `'P'` and `'F'`, respectively.

In case the mesh structure contains specific items (e.g., ridges, corners, normals, tangents), they can also be visualized on the screen. Corresponding keyboard shortcuts are the following : `'g'` for the geometric items (sorry, no possibility of choosing only corners or ridges), `'n'` for the normals and tangents.

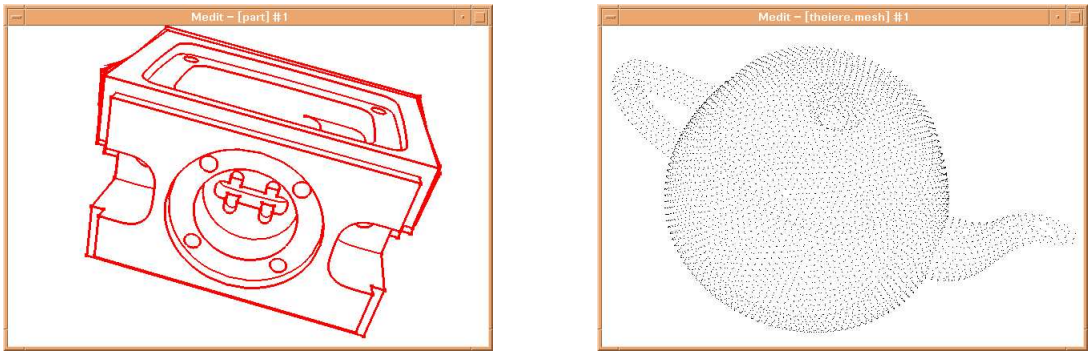


Figure 7: *Visualization of the ridges and corners on a surface triangulation (left-hand side) and visualization of a point cloud (right-hand side).*

A special case occurs if the mesh structure contains only vertices (no elements, *i.e.*, no topology). In this case, the corresponding point cloud is displayed when selecting the mode `'g'`(Figure 7, right-hand side).

5.3.3 Clipping and cutting

Now we turn to the next menu presenting the *Clipping* options. Obviously, this menu deals with cutting planes and related issues. In `MEDIT` a single cutting plane can be defined and activated at any time. To activate the plane, type `F1`, it will be materialized on the screen as a green rectangle slicing the model. By default, the initial plane is $x = x_G$, x_G being the x-coordinate of the mesh barycenter. The plane equation is written at the bottom of the screen.

The color of the clip rectangle indicates the current status of the plane : green for active, magenta for edited, blue for freezed. Keyboard shortcuts `F2` and `F3` allows the user to respectively edit and freeze the plane. When a plane is being edited, any mouse movement (left and middle buttons) will modify the plane equation (in a very similar way the mesh is manipulated). If the plane is freezed, its representation remains unchanged on the screen, however its equation changes as the mesh rotates. This mode can be combined, for instance, with the *automation* mode (see paragraph 5.2.2).

3D elements. If the initial mesh contains 3D elements (tetrahedra or hexahedra), the elements intersected by the plane are made visible (see what happens when cutting the demo example). This feature is especially usefull to fully appreciate the sizes of the elements intersected (Figure 8, left-hand side). This option is primarily intended for people concerned with volumetric meshing problems. If you are more concerned by getting a nice planar slice cut, select the option *Toggle capping* from the *Clipping*

menu. To see only the surface elements (*i.e.*, to remove the 3d entities), select the option *Toggle Vclip* or use key `F4`.

Remark 5.4 *If additional information is associated with the mesh entities, this information will be also processed when a cutting plane is activated (e.g. a scalar map or isolines).*

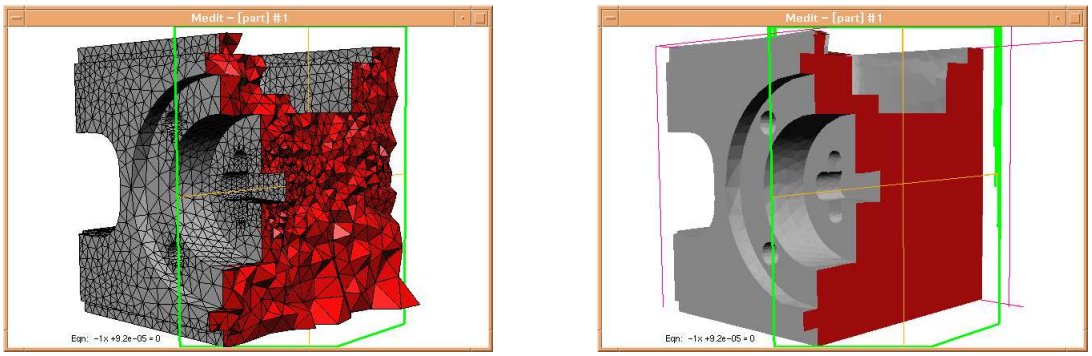


Figure 8: *Example of a cutting plane through a volumetric mesh (left-hand side) and with option* Capping *enabled (right-hand side).*

5.3.4 Picking and selecting

As `MEDIT` is intended for mesh visualization and can be seen, to some extent, as a debugging tool, a fundamental requirement is to be able to designate or pick a mesh entity using the mouse. For various practical reasons, only 2D and 3D entities can be selected (*i.e.*, no vertices or edges).

To select an entity, place the mouse cursor over the entity and click on the mouse left button while pressing the `Shift` key simultaneously. The selected item will be displayed on the screen in a color related to the associated sub-domain and, concurrently, information about this item will be printed on the shell window.

Remark 5.5 *If an item has been selected, by typing* `'V'`*, its barycenter will become the center of rotation. This is especially usefull when zooming on a small part of the mesh, to locally manipulate the mesh.*

Item identification. The reverse operation consists in finding a given item without knowing its relative position on the screen (and in the meshh structure). For instance, if the user wants to see face number 100, he has to type `'#'` in the graphic window and to enter the desired entity number in the shell (text) window as prompted :
`ENTITY NUMBER:`
If the number corresponds to a valid mesh entity (face or element), the corresponding item will be selected and displayed in a different color on the screen. At the same time, the result of the selection is displayed in the shell window :

```
Picking result :
 Triangle  9385 : 2564, 4980, 6515    ref : 508 [MAT12]
   normal : 0.000000 1.000000 0.000000
 vertex    2564 : 0.017155 0.033333 -0.012174   ref 0
 vertex    4980 : 0.017452 0.033333 -0.016508   ref 0
 vertex    6515 : 0.012759 0.033333 -0.013018   ref 0
```

5.3.5 Hiding sub-domains

If the mesh data structure contains several sub-domains (remember that a sub-domain is defined from the reference number associated with a mesh face or a mesh element), it is possible to hide a selected sub-domain. By definition, a sub-domain is selected if a face (or an element) that belongs to this sub-domain is selected (see previous section). To hide a sub-domain type `'r'` and conversely, type `'R'` to restore (show again) a hidden sub-domain. Multiple sub-domains can be hidden and restored at any time.

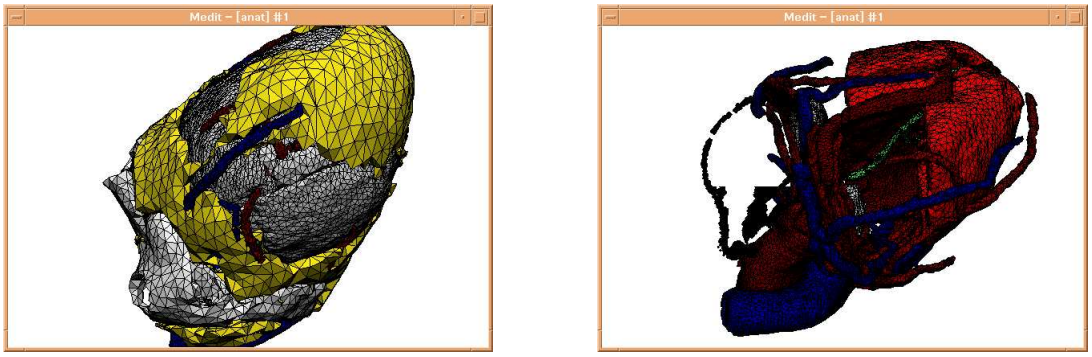


Figure 9: *Example of hiding sub-domains to have a better understanding of a complex structure (here a biomedical organ).*

5.3.6 Visualizing terrains

Using `MEDIT`, the user can visualize terrains or cartesian surfaces using a "flight simulator" mode. This mode is automatically selected after reading a `.gis` file. However, the user can switch between the normal mode and the terrain mode by typing `'J'` at any time. Notice that the viewing matrices of both modes are not directly related.

In this mode, the vertical and horizontal displacements (*i.e.*, elevation and azimuth) are controlled by the mouse. If the automation mode is activated (`'a'`), the speed of displacement can also be controlled using `'-'` and `'+'` keys (a negative speed allows the user to go in the opposite direction). To toggle the information panel, type `'j'`.

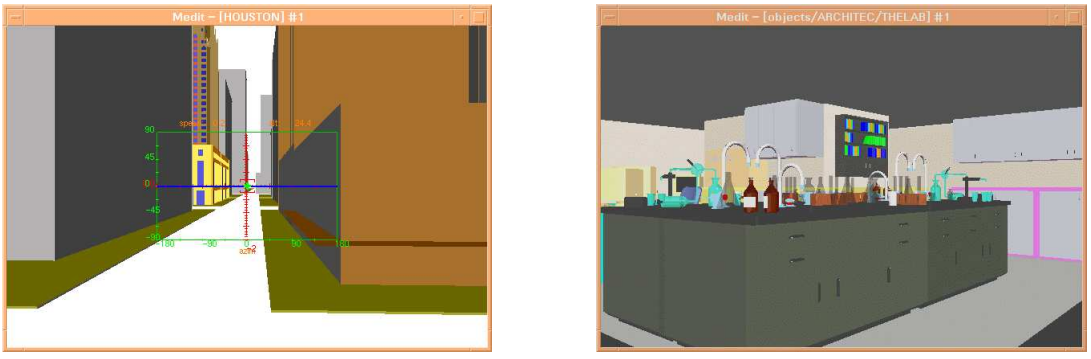


Figure 10: *Examples of terrain visualization : image of a town main street (left-hand side) and interior of a research laboratory (right-hand side).*

5.4 Advanced features

MEDIT offers also more complex features that will be described in this section. In particular, the management of multiple windows and the visualization of scalar fields associated with 2D meshes are presented.

5.4.1 Window splitting

When this option is selected (*Toggle splitview* in *Features* menu), the graphic window is decomposed into 4 sub-windows, corresponding to front, up, down and side view (projections onto xOz, xOy and yOz). The upper right window is a reduction of the main graphic window, all options and features are still available in this small portion of the screen.

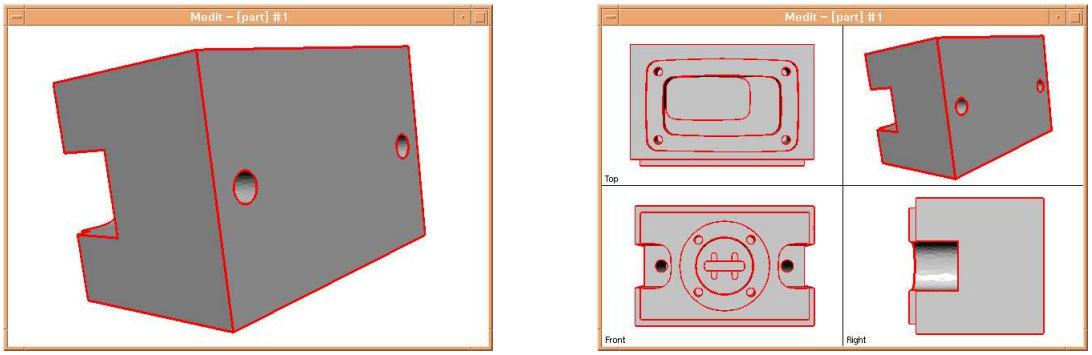


Figure 11: *Example of sub-window activation : original view (left-hand side), split view option (right-hand side).*

5.4.2 Managing multiple views

The main principle of MEDIT is to associate one mesh structure with one graphic window, *i.e.*, if multiple mesh files are supplied on the command line, multiple windows will be opened simultaneously. By default, all views are independent, the corresponding meshes can be manipulated and their attributes can be changed independently. However, in some cases, it may be interesting to link multiple views together, in order to ease the comparison between the meshes. To this end, first select a 'master' view : put the mouse cursor in a graphic window and type `Alt 'c'` to copy the view parameters in a buffer (or select *Copy* in the *View* sub-menu). Then, put the cursor in another graphic window and type `Alt 'l'` to link this view to the previous one (or select *Link* in the menu). Now, when rotating the mesh in the 'master' window, the mesh in the 'slave' window should follow the same transformations. However, the transformations in the 'slave' window are not applied on the 'master' window, the 'slave' view is only affected by a transformation of the 'master' view. You can redo the same operation with another window. Notice however, that you always need to define a pair (master, slave) (*i.e.*, to link multiple windows to a single master window, you need to do copy and link windows in sequence). Instead of linking views together, you can simply copy `Alt 'c'` and paste `Alt 'p'` the current view parameter to a window. At any time, you can return to the initial situation, *i.e.*, un-link the 'master-slave' configuration, by typing `Alt 'u'`.

The option *Duplicate* (`Alt 'd'`) allows to duplicate the current view : a new graphic window is created and the current mesh structure is displayed in both windows. In this mode, the two structures are independent to each other. To close a graphic window, simply type `'X'`, or select *Close window* in the main menu.

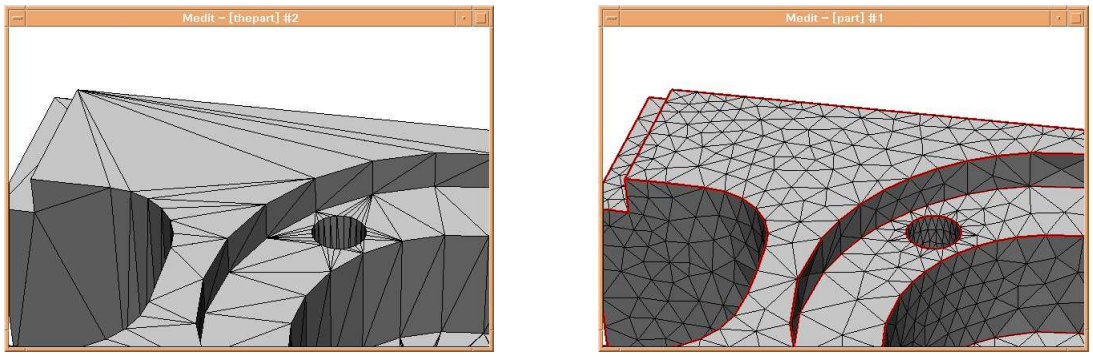


Figure 12: *Multiple views linked together to ease the comparison. Left-hand side : main 'master' view (initial mesh, local enlargement), right-hand side : 'slave' view (optimized mesh, enlargement of the same area).*

5.4.3 Cartesian surfaces

This section concerns the particular case of 2D meshes and associated solutions. In other words, you should have a 2D mesh structure (*i.e.*, a `mesh` file) and a `.bb` file containing scalar values associated with the mesh nodes.

`MEDIT` offers the option of looking at the scalar field as a height map (Figure 13). To this end, simply select the option *Toggle elevation* from the menu *Data*. Within the same menu, you can change the elevation coefficient to any scalar value (as prompted in the shell window). Notice, that this cartesian surface will be considered as a 3D object, thus meaning that you can manipulate it using the mouse or change its color attributes, etc. Typing `'m'` will display the surface using a color map (see Section 5.4.6).

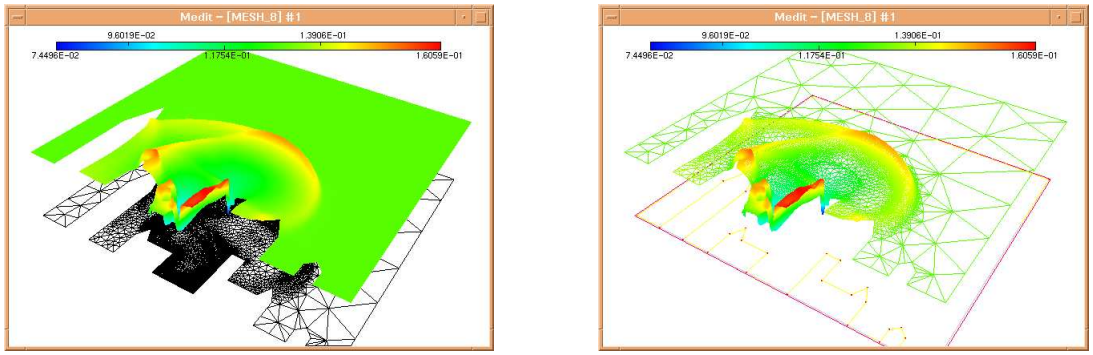


Figure 13: *Example of visualization of a cartesian surface constructed from a scalar map (supplied by an* a posteriori *error estimate) associated with an adapted mesh.*

5.4.4 Output files

`MEDIT` provides a simple mechanism to saving a view. Notice however, that only bitmap images can be written out (*i.e.*, a hardcopy of the screen).

Image file. To save a bitmap image of the screen, select option *Hardcopy PPM* in the menu *File*. By default, the output file format is `ppm` (portable pixmap), a very simple format (supported by many

graphical and image processing tools). Notice that the size of the image (in pixels) matches the current window size. As a consequence, the size of an output image cannot exceed the screen resolution.

Postscript bitmap files. `MEDIT` also offers the possibility of exporting postscript files. The size (in cm) as well as the resolution (in dots per inch) of the file can be specified in the configuration file (see Section 6). Three types of postscript files are valid, corresponding to black and white, greyscale and full color images. For instance, the user can export a postscript file of 20 cm at 300 dpi in full color mode. The size of the output file is related to the quality (resolution) as well as to the output mode : a full color file is much larger than a black and white image.

Remark 5.6 *In the postscript mode, the size of the graphic window is not directly related to the size of the output file.*

5.4.5 Editing materials

In some applications, in order to achieve more realistic rendering (Figure 14, right-hand side), it may be desirable to adjust the color parameters related to the sub-domains. With `MEDIT`, each color can be set very easily in an interactive manner. To this end, pick a mesh entity (face) corresponding to the sub domain you want to edit. Then, type `'E'` (or select option *Edit matcolors*) to display the material editor (Figure 14, left-hand side). This editor consists of a graphic window in which the material color properties are presented as numerical values, the result being displayed on a unit sphere. The user can adjust any of the parameter values using the mouse. Simply put the cursor on a value and move the mouse up and down while pressing the left button until the correct value is set. The corresponding material color is displayed on the sphere. To validate the final value type `'a'` (*i.e.*, to apply this choice on the model). Type `'q'` to quit this editor. The user can adjust the ambient, diffuse, specular, and emission vector values (in rgb mode). Notice that the fourth component of the color vector correspond to a transparency index (1 for opacity, 0 for translucid).

5.4.6 Scalars, vectors, etc.

If a `.bb` file has been sucessfully loaded, it is possible to visualize the scalar/tensor values using a color map.

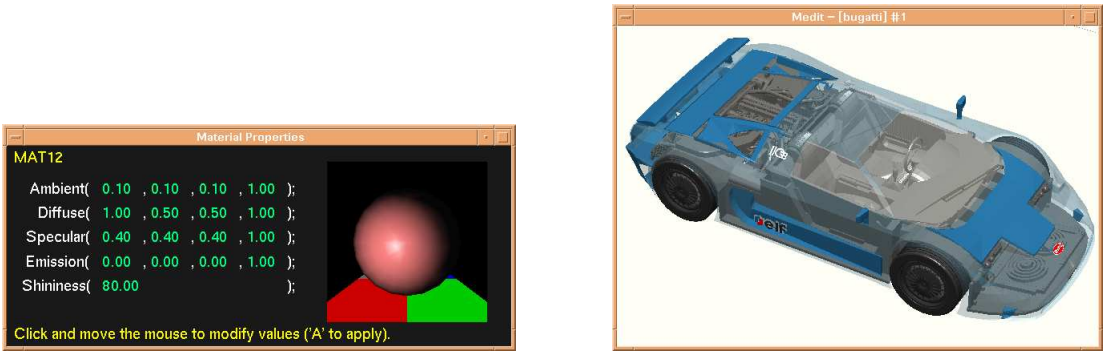


Figure 14: *Material edition window (left-hand side) and example of 'realistic' rendering using multiple materials, including transparency effects (right-hand side).*

Color map. To visualize a discrete scalar field (*i.e.*, associated with mesh points or mesh faces), simply type `'m'` or select *Toggle metric* in the *Data* menu. In this mode, a set of color values is used to represent the range of the numerical values of the data (Figure 15, left-hand side). By default, "cold" colors (dark blue) are used to represent low values, and "hot" colors (orange, red) are used to represent high values. The user can adjust the color index values in the configuration file (cf. Section 6). This option is usefull when looking at multiple mesh structures, in order to use the same color map in all windows (e.g. for comparison purposes). The color palette can be shown/hidden by typing `'p'`.

Remark 5.7 *The choices offered in the* Data *menu depend on the type of the metric values associated with the mesh structure.*

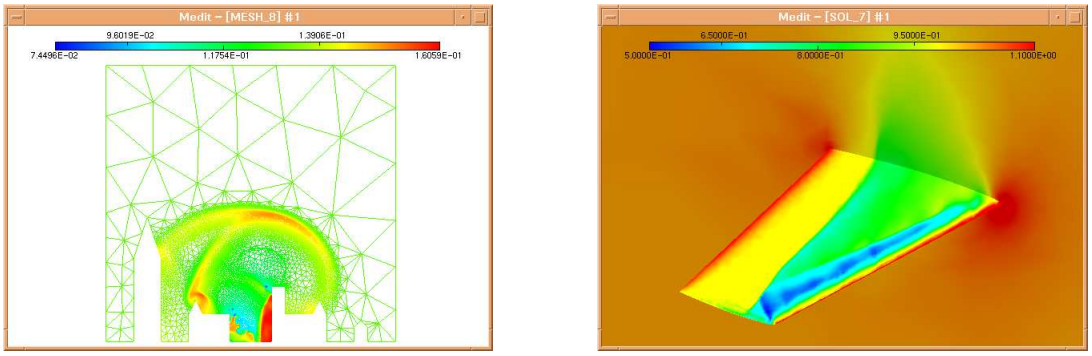


Figure 15: *Example of scalar map visualization in 2D (left-hand side) and 3D (right-hand side).*

Iso-lines, iso-surfaces. For 2D (resp. 3D) data sets, isolines (resp. isosurfaces) are useful for subdividing the data structure into bounded regions corresponding to iso-values. Within the *Data* menu, select *Toggle iso-lines* to visualize the lines of isovalues of the dataset (Figure 16). Similarly, if the mesh is a 3D mesh (containing tetrahedra or hexaedra), isosurfaces can be build and visualized by chosing *Toggle iso-surfaces*. The isosurfaces correspond to the 5 index values defined in the default color palette. The user can adjust the values of the isolines or isosurfaces in the configuration file.

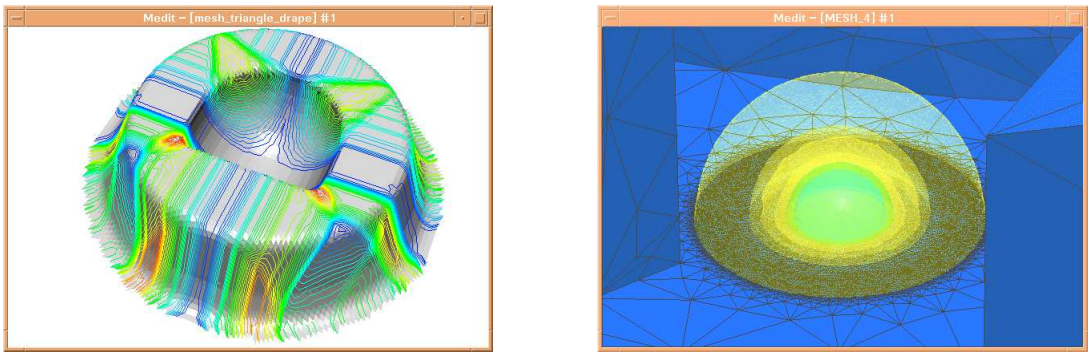


Figure 16: *Example of isolines (left-hand side) and isosurfaces (right-hand side) reconstruction from surface and volume meshes.*

Streamlines. This module is only available if a vector field is defined at the mesh vertices. The *streamlines* option is used for vector field visualization : the user can visualize local flow phenomena in critical regions (e.g. vortices and turbulence in CFD computations). The path of particles are computed through the vector field and represented as discrete lines, each corresponding to a streamline. The line follows a particle path, its color is adjusted based on the variation of the module of the vector (Figure 17).

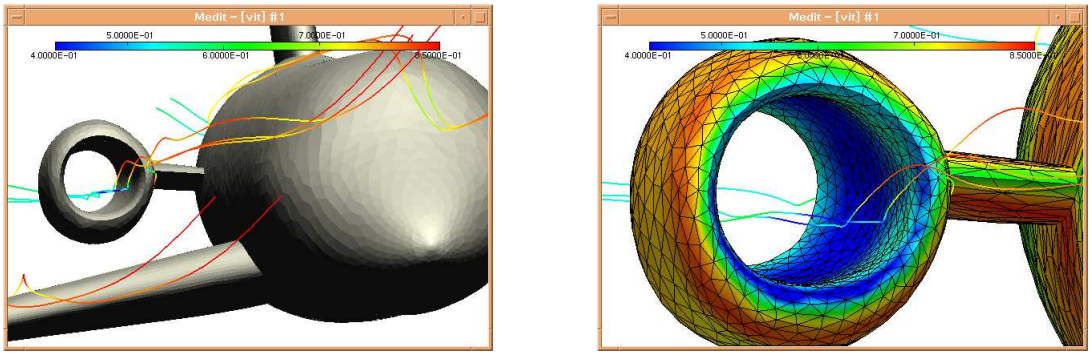


Figure 17: *Example of streamlines in a CFD computation.*

From a practical point of view, the user must specify the streamlines in a separate `.iso` data file. The structure of this file is very simple, it is composed of two fields identified by keywords :

- `NbLines`, followed by :
  - an integer corresponding to the number of desired streamlines and
  - a series of point coordinates $x_i\ y_i\ z_i$ representing the starting points of the streamlines.
- `Box`, followed by a series of 6 scalar values representing the bounding box of the streamlines (to truncate the lines) : $x_{min}\ x_{max}\ y_{min}\ y_{max}\ z_{min}\ z_{max}$.

The following lines illustrate the structure of this file :

```
NbLines
2
13. 1.4 0.2
13. 1.6 0.2

Box
-10 30
-20 20
-20 20
```

In this example, two lines have been defined, starting from the points $(13, 1.4, 0.2)$ and $(13, 1.6, 0.2)$ and the domain has been truncated by a parallelepiped.

5.5 More advanced features

In this section, slightly more complex features of MEDIT are described. We recommend the user to be familiar with the concepts introduced in the previous sections first, before reading this section.

5.5.1 Animations - file sequences

Here, we consider the case where a sequence of meshes (and possibly associated values) have been computed and stored. A typical field of application concern the mesh adaptation methods in numerical simulations. For instance, to capture a transient phenomenon in CFD, it is (almost) required to adapt the mesh very often in order to capture the shock waves [1]. MEDIT allows the user to visualize the resulting sequence of meshes very easily. Here, the aim is to fix the view parameters on the first mesh of the sequence and then to visualize all meshes in the sequence successively.

At first, the user must save the sequence of meshes (and their associated data) in separate files (in binary or ascii format), under the names : `xxx.001.mesh` to `xxx.253.mesh`. To visualize the sequence of meshes, type :

```
medit xxx -a 1 253
```

The syntax of this command is `-a start stop`, where `start` (resp. `stop`) represents the number of the first (resp. last) mesh in the sequence. Notice that numbers in mesh name must be written using 3 digits (e.g. 001, 012, 129) with a '.' after the mesh prefix, for instance `mesh.012.mesh`.

Similar to the classical case (*i.e.*, where a single mesh is loaded), a graphic window is opened in which only the first mesh of the series is displayed. The user can then select any rendering mode, activate a clip plane, edit the sub-domains colors, adjust the view parameters and manipulate the mesh using the mouse. There is no restriction on the action the user can apply on the mesh.

To start visualizing the whole sequence, select the option *Play Sequence* in the *Animation* menu. The meshes are rendered consecutively, the graphic window being refreshed after a new mesh has been loaded. It is possible to save a color bitmap (in `ppm` format) of each mesh by selecting the *Toggle ImgSave* of the *Animation* menu. This option is useful when creating animated images files (e.g., animated `gif`). For instance, on Unix/Linux systems, use a tool like 'convert' to produce such images :

```
convert xxx*.ppm xxx.gif -delay 20
```

this will generate a single animated `gif` image.

Remark 5.8 *In the animation mode, all meshes are considered independent from each other, in terms of the number of mesh entities. Only the graphic attributes are fixed by the user at the beginning of the sequence.*

Remark 5.9 *The mouse cursor must stay in the window while creating the animation sequence.*

5.5.2 Morphing

This feature can be considered as a peculiar case of the animation mode : the morphing sequence is composed of two meshes only, the initial and final meshes having exactly the same number of mesh vertices (the vertices are also supposed to be matched).

To load a morphing sequence, type : `medit -m mesh1 mesh2`. A graphic window is opened, displaying `mesh1` the first mesh of the sequence. To visualize the following meshes of the sequence,

simply type `'M'`, iteratively (cf. Figure 18). From the technical point of view, a simple linear interpolation scheme is used on the mesh vertex coordinates to compute each mesh in the sequence (a sequence is composed of 100 meshes). By selecting the option *Toggle Imgsave* in the *Animation* menu, the user can save a whole series of bitmap files, and eventually create an animated `gif` image (see previous section).

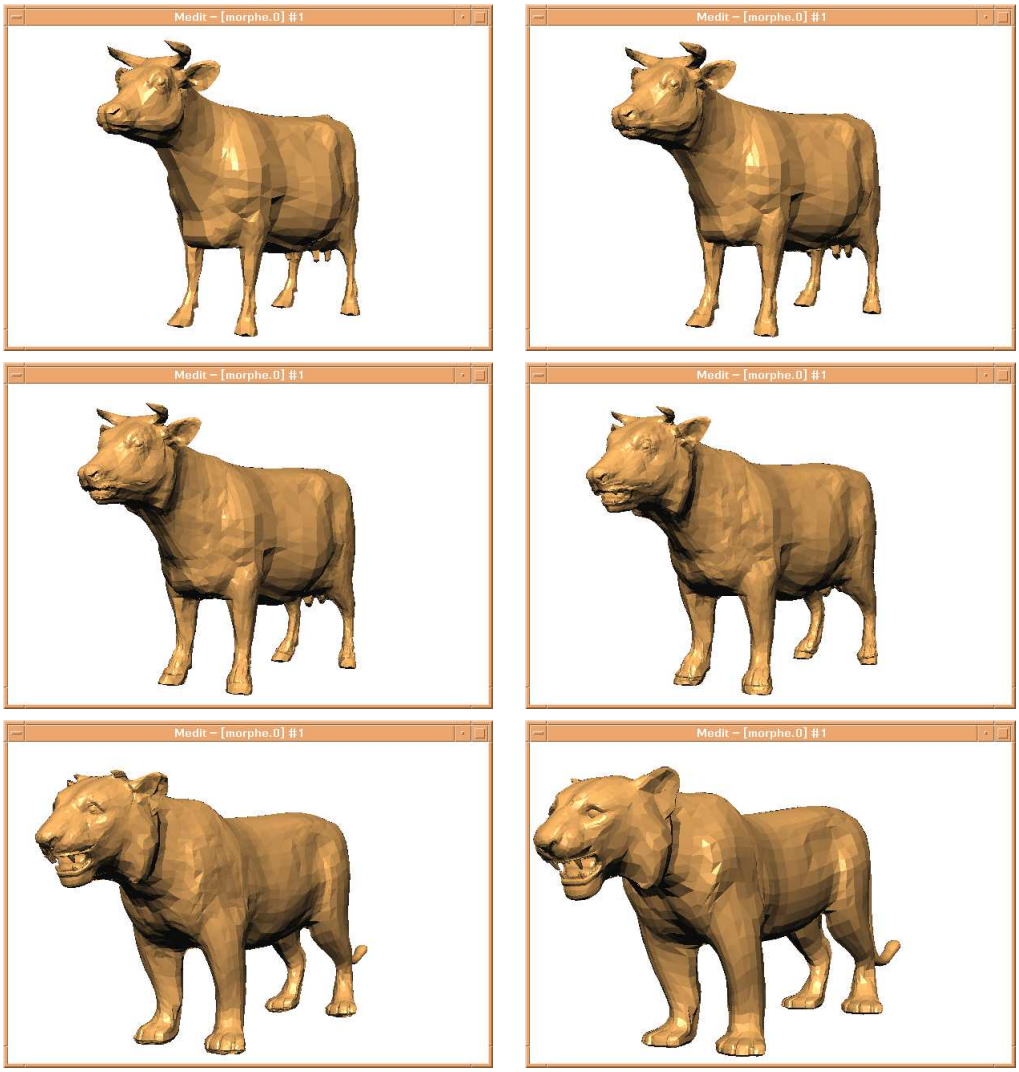


Figure 18: *Example of a morphing sequence (meshes at iterations 1, 10, 20, 30, 40 and 50).*

6 Customizing MEDIT

In addition to the various menu options, additional parameters can be stored in a simple text file, named `xxx.medit` and associated with the `xxx.mesh` file. The file structure is organized as a series of fields, composed of a keyword and scalar values. In case no `xxx.medit` file is found, `MEDIT` is looking for a `DEFAULT.medit` file in the current directory.

Remark 6.1 *All parameters have been assigned reasonable default values, thus making the specification of such file not strictly required.*

File structure. Parameters that can be specified are the following (the keywords are not case sensitive) :

- `BackgroundColor`, $r\ g\ b$
  indicates the color background components (single precision floats) ranging from 0 to 1, default is $(0, 0, 0)$ (black),

- LineColor, $r\ g\ b$
  specifies the color of the mesh edges (single precision floats), default is $(1, 1, 1)$ (white),

- `SunPosition`, $x\ y\ z$
  indicates the "light" source position (single precision floats), default is $(x_{eye}\ y_{eye}\ z_{eye})$,

- `WindowSize`, $w\ h$
  initializes the window size (2 integers, width, height) in pixels,

- `RenderMode`, str
  this string specifies the default rendering mode, str can be any of the following keywords : `hidden`, `fill`, `colorshading`, `shading`,

- `Palette`, $c_i\ i = 1, 5$
  interpolation values used for color map rendering (sinle precision floats). c_1 and c_5 are used to truncate the associated scalar values. The intermediate values are also used when building iso-lines or iso-surfaces.

- `Postscript`, $cm\ dpi\ k\ str$
  used when creating a postscript file, cm specifies the size in centimeters, dpi corresponds to the desired $dotperinch$ resolution (default is 300) and k is a coefficient used (if not null) to brighten slightly the image colors. In the current version, str is not used.

A comment line starts with a # character and ends at the end of the line. Comments can occur only between fields.

The user can edit the parameters values directly in the configuration file using a simple text editor and re-load the new parameters by selecting the option *Load prefs* from the *File* menu. Similarly, current parameter values can be written on disk by chosing the option *Save prefs* from the *File* menu.

Material properties. When using lighting effects, polygon are not having a single particular color, but rather consists in materials having certain specific reflective properties. The user can supply the reflective properties of materials for ambient, diffuse and specular light source. In addition, all materials can emit with a distinct color.

In the configuration file, material properties can be specified. To this end, the user must first indicate the number of materials properties that are provided using the keyword :
`NbMaterial`, n
Then, n materials properties need to be specified as follows :
`Material mat`
$a_r\ a_g\ a_b\ a_t$
$d_r\ d_g\ d_b\ d_t$
$s_r\ s_g\ s_b\ s_t$
$e_r\ e_g\ e_b\ e_t$
s where a, d, s, e represent respectively the ambient, diffuse, specular and emission coefficients. By default, 4 components are required for each material property, the first three correspond to the $r\ g\ b$ color coordinates (ranging in intensity from 0 to 1), the last one being a translucent coefficient (transparency is enabled if $x_t < 1$).

Example of configuration file.

```
#  File created with MEDIT 2.1

BackgroundColor
1. 1. 1.
WindowSize
600 400
postscript
8. 300. 0. color

NbMaterials
2
Material DEFAULT_MAT
0.100000 0.100000 0.100000 1.000000
0.800000 0.800000 0.800000 1.000000
0.400000 0.400000 0.400000 1.000000
0.000000 0.000000 0.000000 1.000000
80.000000
# Material MAT01
Material MAT01
0.100000 0.100000 0.100000 1.000000
1.000000 0.000000 0.000000 1.000000
0.400000 0.400000 0.400000 1.000000
0.000000 0.000000 0.000000 1.000000
80.000000

End
```

7 Appendix

This section provides some information about the error messages and file formats used in the software. The list of keyboards shortcuts is also given in this section. At the end of this section a list of Frequently Asked Questions will help the user to better understand the main options and features of MEDIT.

7.1 List of error messages

Warnings and error messages are identified and listed hereafter. Usually, the syntax of an error message is the dollowing :
`ERR xxxx, proc, MESSAGE, list`
where xxx stands for the error number, `proc` is the name of the procedure in which the error occured, `MESSAGE` is the error diagnostic and `list` is a facultative list of arguments related to this error (e.g., a list of mesh entities). The first digit of the error number indicates in which stage of the process the error occured :

0 preliminary stage (data reading),
1 display list creation,
2 output problem (unable to save file, etc.),
9 runtime dependent problem : cancelled job, hardware problem, etc.

The list of diagnostics is as follows :

[L0] : Level 0 : input data related errors.
These errors mainly occurs because of incorrect or corrupted data file(s) or memory allocation problems.

ERR	MESSAGE	DIAGNOSTIC / CORRECTION
0000	UNABLE TO OPEN FILE	Check the file permissions
0001	DATA FILE NOT FOUND	Check file name
0002	NO INPUT DATA	No mesh entity, check mesh data file
0003	WRONG DATA TYPE	Wrong dimension
0004	INCORRECT KEY WORD	Check spelling
0010	DATA DISCARDED	Data not recognized
0011	DATA TYPE DISCARDED	Category of mesh entity discarded. Check spelling

[L1] : Level 1 : display list creation.

ERR	MESSAGE	DIAGNOSTIC / CORRECTION
1000	UNABLE TO ALLOCATE MEMORY	Mesh structure requires more memory. Upgrade worsktation capabilities.
1001	MAX VALUE EXCEEDED	Too many mesh names supplied.
1002	GRAPHIC RESSOURCE MISSING	Unable to open graphic window

[L2] : Level 2 : output file problems.

ERR	MESSAGE	DIAGNOSTIC / CORRECTION
2000	UNABLE TO SAVE FILE	Check the file permissions.

[L9] : Level 9 : errors related to software problems.

ERR	MESSAGE	DIAGNOSTIC / CORRECTION
9904	PROGRAM KILLED BY USER	
99xx	FATAL ERROR ENCOUNTERED	Segmentation fault, bus error, etc. Contact hot line.

7.2 File formats

As pointed out in Section 4.2, The mesh data structure can be described using very simple (although complete) data formats. By default (*i.e.*, if no extension is supplied), the program will first attempt to find a `mesh` file (in binary format). If no such file is found, then other file formats will be tried, successively the `msh2` format then the `gis` format.

7.2.1 The `mesh` format

This format is composed of a single (binary or text) data file. Its structure is organized as a series of fields identified by keywords. The blanks, "newline" or <CR> and tabs are considered as item separators. A comment line starts with the character # and ends at the end of the line. The comments are placed exclusively between the fields.

The mesh file must start with the descriptor :

```
MeshVersionFormatted  1
Dimension   3
```

The other fields supported by `MEDIT` are either required or faculative. The required fields correspond to the geometry (*i.e.,* the coordinates) and to the topology description (*i.e.,* the mesh entities). In the following tables, the term v_i indicates a vertex number (*i.e.*, the i^{th} vertex in the vertex list), e_i is an edge number, t_i is a triangle number and q_i is a quadrilateral number. Notice that the vertices coordinates are real numbers in single precision.

Keyword	Card.	Syntax	Range
`Vertices`	np	$x_i\ y_i\ z_i\ ref_i$	$\{i = 1, np\}$
`Edges`	ne	$e_i^1\ e_i^2\ ref_i$	$\{i = 1, ne\}$
`Triangles`	nt	$v_i^j\ ref_i$	$\{i = 1, nt\}$, $\{j = 1, 3\}$
`Quadrilaterals`	nq	$v_i^j\ ref_i$	$\{i = 1, nq\}$, $\{j = 1, 4\}$
`Tetrahedra`	$ntet$	$v_i^j\ ref_i$	$\{i = 1, ntet\}$, $\{j = 1, 4\}$
`Hexaedra`	nh	$v_i^j\ ref_i$	$\{i = 1, nh\}$, $\{j = 1, 8\}$

Then, follows the description of constrained entities or singularities. In particular, a corner (keyword `Corner`) is a C^0 continuity point (this type of item is necessarily a mesh vertex). By analogy, a `Ridge` is an edge where there is a C^0 continuity between the adjacent faces. The fields of type `Requiredxx` make it possible to specify any type of entity that must be preserved by the meshing algorithm.

Keyword	Card.	Syntax	Range
`Corners`	nc	v_i	$\{i = 1, nc\}$
`RequiredVertices`	nrv	v_i	$\{i = 1, nrv\}$
`Ridges`	nr	e_i	$\{i = 1, nr\}$
`RequiredEdges`	nre	e_i	$\{i = 1, nre\}$

As mentioned above, it is also possible to specify normals and tangents to the surface. The normals (resp. tangents) are given as a list of vectors. The normal at a vertex, keyword `NormalAtVertices`, is specified using the vertex number and the index of the corresponding normal vector. The normal at a vertex of a triangle, `NormalAtTriangleVertices`, corresponds to the combination of the triangle number, the index of the vertex in the triangle and the index of the normal vector at this

vertex. Similarly for the field corresponding to the keyword `NormalAtQuadrilateralVertices`. The tangent vectors are described in the same way.

Keyword	Card.	Syntax	Range
`Normals`	nn	$x_i\ y_i\ z_i$	$\{i = 1, nn\}$
`Tangents`	nnt	$x_i\ y_i\ z_i$	$\{i = 1, nnt\}$
`NormalAtVertices`	nv	$v_i\ n_i$	$\{i = 1, nv\}$
`NormalAtTriangleVertices`	ntv	$t_i\ v_j\ n_i$	$\{i = 1, ntv\}$
`NormalAtQuadrilateralVertices`	nqv	$q_i\ v_j\ n_i$	$\{i = 1, nqv\}$
`TangentAtEdges`	te	$e_i\ v_j\ t_i$	$\{i = 1, te\}$

Finally, the data structure must end with the keyword : `End`.

7.2.2 The `msh2` format

This file format is a very crude data format provided for backward compatibility purposes (the user is strongly encouraged to use the `mesh` format). Notice that only surface meshes can be described with this format. It is composed of two ASCII files, `xxx.points` and `xxx.faces`. The file `.points` has the following structure :
np
$x_i\ y_i\ z_i\ ref_i$
representing the vertex coordinates in single precision and the vertex reference. The file `.faces` represents the mesh topology and contains the following records :
nf
$d\ v_i^1\ v_i^2\ v_i^3\ ref_i\ ref_i^1\ ref_i^2\ ref_i^3$
where d is either 3 (triangles) or 4 (quad), v_i represents a vertex number, ref_i is the reference of the sub-domain containing the face and ref_i^j are the edge references.

7.2.3 The `gis` format

This file format is used to import terrains (or cartesian surfaces). It consists of a single file (ASCII or binary). The file structure is composed of a header (ASCII) indicating the terrain resolution and spatial location :
G_i
$sx\ sy$
$cx\ cy\ cz$
$gu\ hu$
$xmi\ ymi$
where G_i indicates the file type (G_1 for an ascii file and G_2 for a binary file), $sx\ sy$ are two integers defining the terrain resolution (*i.e.*, the size of the underlying grid), $cx\ cy\ cz$ represent 3 scaling factors, $gu\ hu$ are the model units and $xmi\ ymi$ represent the location of the lower left corner of the terrain. Then, follows a series of scalar (single float) values figuring the heights of each node of the grid.

7.2.4 The `bb` format

This ASCII file contains scalar/tensor values associated with mesh entities. This file (`xxx.bb`) has the following structure :
$dim, nbmet, nbval, type$

v^i, $i \in [1..nbmet \times nbval]$
where `dim` is the dimension of the space, `nbmet` the number of related fields. For instance, `nbmet=1` corresponds to scalar values, `nbmet=3` and `dim=2` defines a 2×2 symetrical matrix. `nbval` indicates the number of information attached to the vertices (`type=2` and `nbval=np`) or to the faces or elements (`type=1`). Then, a second record contains the $nbmet \times nbval$ values associated with mesh entities. For instance,

```
3 1 209 2
7.76059 8.565 9.58969 7.76059 9.58969 8.565 8.46321 8.46321
11.05018 10.36006 11.05018 8.74209 8.74209 11.15122 11.15122
...
```

represents a scalar field defined at the vertices of a surface mesh.

7.3 Glossary

In this section, we recall the keyboard shortcuts that are defined in MEDIT. They can be used at any time during the visualization to toggle mesh attributes or software features.

7.3.1 Options by feature

- *Rendering options* :
  notice that all these options work on a ON/OFF mode ('toggle' options);

Key	attribute	key	attribute	key	attribute
f	facets	l	lines	g	entities
c	object color	e	material color	b	back. color
A	axis	B	box	G	grid
C	capping	r	hide subdom.	R	show subdom.
n	smooth shading				

- *Control options* :

Key	attribute	key	attribute	key	attribute
i	reset view	a	animate	I	interactive
h	online help	q	quit	X	close window

- *Viewing options* :
  these options are invoked by typing a character while pressing the ALT,

Key	attribute	key	attribute	key	attribute
c	copy	d	duplicate	p	paste
l	link	u	unlink		
J	flight mode (toggle)	y	change up axis		

- *Misc. features* :

Key	attribute	key	attribute	key	attribute
L	load prefs	W	save prefs	H	hardcopy PPM
F	face nums	P	point nums	#	select entity
N	normals	O	oppos. normals		
m	data	o	iso-lines	w	tensor/vector
k	elevation	K	elev. coeff.		
j	print info	p	palette		
+/-	scale object	z/Z	scale view		
F1	clip ON	F2	edit clip	F3	freeze clip
F4	toggle Volclip				
F5	shrink	F6	increase shrink	F7	decrease shrink

7.3.2 Options in alphabetic order

The following tables show the keyboard shortcuts in alphabetic order.

- *lowercase single keys :*

key	feature	toggle	default
a	animate	x	OFF
b	back color	-	
c	object color	x	ON
d			
e	material color	x	OFF
f	show facet	x	ON
g	specified entites (corners, ...)	x	OFF
h	help on-line	-	
i	reset view	-	
j	show info (palette, plane eqn, ...)	x	ON
k	elevation map	x	OFF
l	show polylines	x	ON
m	color map	x	OFF
n	smooth shading (if normals given)	x	ON
o	iso-lines	x	OFF
p	palette	x	ON
q	quit Medit	-	
r	hide reference	-	
s	hide picked item	-	
t			
u			
v			
w	tensor/vector map	x	OFF
x			
y	change up axis (flight mode)	-	
z	zoom in (change fovy)	-	

- *combination of keys* : `ALT key` (if supported by system)

key	feature
c	copy view parameters (select window)
d	duplicate view in new window
l	link active view to the selected window
p	paste selected view parameters in active window
u	unlink view

- *rubberband* : `Ctrl` + left button of the mouse

- *uppercase single keys* :

key	feature	toggle	default
A	axis	x	OFF
B	bounding box	x	OFF
C	capping (if clip ON)	x	OFF
D			
E	edit material properties	-	
F	face number	x	OFF
G	show grid	x	OFF
H	hardcopy (bitmap PPM)	-	
I	show object when rotating	x	ON
J	flight mode (dim 3 only)	x	OFF
K	enter elevation coeff.	-	
L	load prefs from file	-	
M			
N	display normal vectors	x	OFF
O	invert normal orientation	-	
P	point numbers	x	OFF
Q			
R	unhide reference	-	
S			
T			
U			
V	change center of scene (picked item)	-	
W	save prefs to file	-	
X	close window (quit Medit if 1 window)		
Y			
Z	zoom out (change fovy)	-	

- *misc. single keys :*

key	feature
+/-	scale object (in / out)
+/-	increase/reduce speed if animate inflight mode
#	enter entity number (facet or point)
F1	clip ON
F2	edit (modify interactively) clip plane
F3	freeze clip plane
F4	toggle Volume clip (tets or hexas)
F5	toggle shrink
F6/F7	increase/decrease shrink value
←→↑↓	translate object

References

[1] F. ALAUZET, P.L. GEORGE, B. MOHAMMADI, P.J. FREY AND H. BOROUCHAKI (2001), Transient fixed point based unstructured mesh adaptation, *ECCOMAS Computational Fluid Dynamic Conf.*, Swansea, UK.

[2] P.J. FREY AND P.L. GEORGE (2000), Mesh generation. application to finite elements, Hermès Science Publ., Paris, Oxford, 814 pages.

[3] P.J. FREY (2000), Medit : outil de visualisation interactif, RT-INRIA 0247, janv.

[4] P.J. FREY (2001), Yams : A fully Automatic Adaptive Isotropic Surface Remeshing Procedure, RT-INRIA 0252, nov.

Index

R

S

T

U

W

Z

Unit´e de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifi que,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399