ovisumesh Package User's Guide, version 0.0.13*

François Cuvelier† Gilles Scarella ‡

Tuesday 29th November, 2016

Contents

*Compiled with Octave [4.0.1]

†Université Paris 13, Sorbonne Paris Cité, LAGA, CNRS UMR 7539, 99 Avenue J-B Clément, F-93430 Villetaneuse, France, cuvelier@math.univ-paris13.fr

‡Université Côte d'Azur, CNRS, LJAD, F-06108 Nice, France, gilles.scarella@unice.fr.

This work was partially supported by ANR Dedales.

Abstract

ovisumesh is an Octave package which allows to handle 2D and 3D simplicial meshes. Mesh loading, mesh visualization and data visualization are enabled. Possible mesh formats are the ones of Triangle (2D), FreeFem++/medit (2D-3D) and gmsh (2D-3D).

1 Introduction

This package allows to read and to handle simplicial meshes generated by gmsh [5], FreeFem++/medit [4, 7] or Triangle [10].

Meshes could be used by image processing, finite element or finite volume computations.

Many functions are issued from *mOptFEMP1* [1] and *mVecFEMP1* [3, 2] packages. (See François Cuvelier's software page.

This package can be used either with Matlab or GNU Octave. It has been tested from Octave 3.8.2 to 4.2.0. It has been tested under Linux (Ubuntu 14.04, 16.04, Debian 7, 8, CentOS 7, OpenSUSE 13.2) and Mac OS X (El Capitan). For Mac OS X, it is only an experimental release as the result of some figures should be improved.

2 Installation of the package

ovisumesh package may be installed as a GNU Octave package or from a typical archive file (of extension .tar.gz, .zip or .7z). Some mesh samples are available to test the package.

2.1 Url of the package

The package can be downloaded at `https://www.math.univ-paris13.fr/~cuvelier/software/VisuMesh.html`.

2.2 Requirements

The package depends on *splines*, *msh* and *general* packages.
Some instructions on how to install Octave and the packages can be found at the following link.

2.3 Installation as a GNU Octave package

Once the package downloaded, in the Octave window the following command allows to install the package

```
pkg install ovisumesh-1.0.0.tar.gz
```

The package can be used in the same Octave session.

2.4 Installation by an archive file (.zip, .tar.gz or .7z)

If the package is downloaded as an archive of type *.tar.gz*, *.zip* or *.7z*, the files need only to be extracted. The user may extract the archive anywhere he wants.

For example, with tar command, in a Linux terminal, in the right directory,

```
tar xvzf ovisumesh-1.0.0.tar.gz
```

2.5 Meshes

Several meshes are used in demos. Meshes could be installed during the installation of the package (during the "pkg install" command) or they could be downloaded here. Then one needs to extract meshes from the archive file *meshes.tar.gz*.

```
tar xvzf meshes.tar.gz
```

3 Initialization

3.1 Method with a GNU Octave package

If the package has been installed as a GNU Octave package, the user has to load the package

```
pkg load ovisumesh
```

3.2 Method with a typical archive file (.zip, .tar.gz or .7z)

```
addpath('<path>/ovisumesh/');
```

The user needs to add the paths of the package subdirectories.

3.3 Mesh samples

(Only for the method with a typical archive file) The user needs to add the path of the mesh samples to his path. For the package installation, meshes are contained in the source files of the package.

4 Reading a mesh

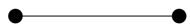
In the package only meshes composed of d-simplices are considered. A mesh structure close to the one in FreeFem++ is used in the package.

4.1 About d-simplices

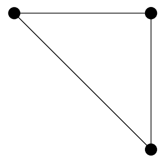
A d-simplex is made of (d+1) vertices. Most common d-simplices are the following:

- A 0-simplex is a node or a vertex.

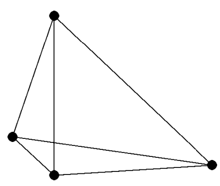
- A 1-simplex is a segment.



- A 2-simplex is a triangle.



- A 3-simplex is a tetrahedron.



4.2 Description of the mesh structure

The data structure associated to a mesh employs many notations already used in FreeFem++ (see [7]). See also the report about the vecFEMP1 package [2].

Mesh structure associated to $\mathcal{T}_h$

d	:	integer simplex dimension
dim	:	integer space dimension
$\mathrm{n_q}$	:	integer number of vertices
$\mathrm{n_{me}}$	:	integer number of elements (d-simplices)
$\mathrm{n_{be}}$	:	integer number of boundary elements ((d−1)-simplices)
q	:	dim-by-$\mathrm{n_q}$ array of reals array of vertex coordinates
me	:	(d+1)-by-$\mathrm{n_{me}}$ array of integers connectivity array for **mesh elements**
mel	:	1-by-$\mathrm{n_{me}}$ array of integers array of mesh element labels
be	:	d-by-$\mathrm{n_{be}}$ array of integers connectivity array for **boundary elements**
bel	:	1-by-$\mathrm{n_{be}}$ array of integers array of boundary element labels
vols	:	1-by-$\mathrm{n_{me}}$ array of reals array of mesh element volumes
h	:	double mesh step size (=maximum edge length in the mesh)
hmin	:	double minimum edge length in the mesh
info	:	two field structure containing the name and the format of the mesh file

More precisely

- $\mathrm{q}(\nu, j)$ is the ν-th coordinate of the j-th vertex, $\nu \in \{1, \ldots, \mathrm{dim}\}$, $j \in \{1, \ldots, \mathrm{n_q}\}$. The j-th vertex will be also denoted by $\mathrm{q}^j = \mathrm{q}(:, j)$.
- $\mathrm{me}(\beta, k)$ is the storage index of the β-th vertex of the k-th element (d-simplex), in the array q, for $\beta \in \{1, ..., \mathrm{d}+1\}$ and $k \in \{1, \ldots, \mathrm{n_{me}}\}$. So $\mathrm{q}(:, \mathrm{me}(\beta, k))$ represents the coordinates of the β-th vertex of the k-th mesh element.
- $\mathrm{be}(\beta, l)$ is the storage index of the β-th vertex of the l-th boundary element ((d−1)-simplex), in the array q, for $\beta \in \{1, ..., \mathrm{d}\}$ and $l \in \{1, \ldots, \mathrm{n_{be}}\}$. So $\mathrm{q}(:, \mathrm{be}(\beta, l))$ represents the coordinates of the β-th vertex of the l-th boundary element.
- $\mathrm{mel}(k)$ is the label of the k-th d-simplex .
- $\mathrm{vols}(k)$ is the volume of the k-th d-simplex .
- info.name is a string of the short name of the mesh file (=relative path in Linux).
  info.format is the format of the mesh file: it can be 'freefem', 'medit' or 'gmsh'.

For example, we give in Figure 1 a ring mesh with 36 vertices, 48 mesh elements and 24 boundary elements. The data structure associated to the mesh is denoted by $\mathcal{T}_h$ and verifies

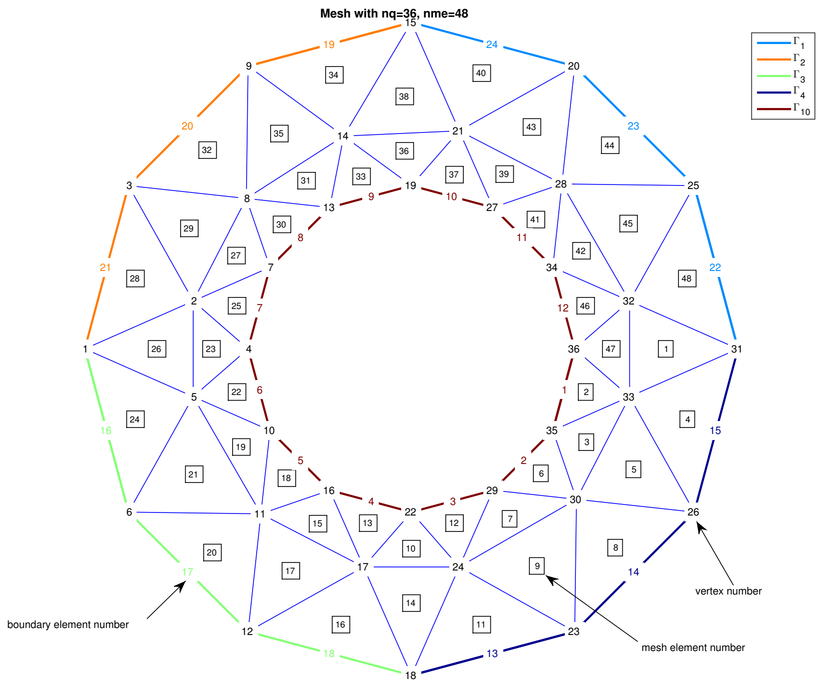


Figure 1: Coarse ring mesh

$$\mathcal{T}_h.\mathrm{n_q} = 36,\ \mathcal{T}_h.\mathrm{n_{me}} = 48,\ \mathcal{T}_h.\mathrm{n_{be}} = 24,$$

$$\mathcal{T}_h.\mathrm{q} = \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & & 34 & 35 & 36 \end{array} \\ \begin{pmatrix} -1.000 & -0.668 & -0.866 & \dots & 0.433 & 0.433 & 0.500 \\ 0.000 & 0.146 & 0.500 & \dots & 0.250 & -0.250 & -0.000 \end{pmatrix} \end{array}$$

$$\mathcal{T}_h.\mathrm{me} = \begin{array}{c} \begin{array}{ccccccccccc} 1 & 2 & 3 & 4 & 5 & & 44 & 45 & 46 & 47 & 48 \end{array} \\ \begin{pmatrix} 33 & 35 & 35 & 31 & 30 & \dots & 25 & 32 & 32 & 33 & 31 \\ 31 & 33 & 30 & 33 & 26 & \dots & 20 & 25 & 34 & 32 & 25 \\ 32 & 36 & 33 & 26 & 33 & \dots & 28 & 28 & 36 & 36 & 32 \end{pmatrix} \end{array}$$

$$\mathcal{T}_h.\mathrm{be} = \begin{array}{c} \begin{array}{ccccccccccc} 1 & 2 & 3 & 4 & 5 & & 20 & 21 & 22 & 23 & 24 \end{array} \\ \begin{pmatrix} 36 & 35 & 29 & 22 & 16 & \dots & 9 & 3 & 31 & 25 & 20 \\ 35 & 29 & 22 & 16 & 10 & \dots & 3 & 1 & 25 & 20 & 15 \end{pmatrix} \end{array}$$

$$\mathcal{T}_h.\mathrm{bel} = \begin{array}{c} \begin{array}{ccccccccccc} 1 & 2 & 3 & 4 & 5 & & 20 & 21 & 22 & 23 & 24 \end{array} \\ \begin{pmatrix} 10 & 10 & 10 & 10 & 10 & \dots & 2 & 2 & 1 & 1 & 1 \end{pmatrix} \end{array}$$

4.3 Supported mesh types

Only few mesh formats are supported by the package. Table 1 details supported mesh formats, the space dimension and mesh file extension. A list of 2D or 3D mesh samples generated by *gmsh* is available online at `https://www.math.univ-paris13.fr/~cuvelier/software/gmshgeo.html`.

	2D				3D	
mesh software mesh file extension	freefem .msh	medit .mesh	triangle —	gmsh .msh	freefem/medit .mesh	gmsh .msh

Table 1: File extension and mesh format

4.4 *GetMeshOpt* function

4.4.1 Description

In the package the function *GetMeshOpt* enables to read a mesh and to construct the mesh structure. Mesh file name and the space dimension are required.

4.4.2 Usage

- Basic usage

```
Th=GetMeshOpt(cFileName,dim);
```

- With all options

```
Th=GetMeshOpt(cFileName,dim,'format',...);
```

4.4.3 Arguments

***cFileName* (input parameter)** is a string which contains the mesh file name. An absolute path is enabled.

***dim* (input parameter)** is an integer which defines the space dimension. For example, if $dim = 3$, then the mesh is a 3D mesh and there are 3 coordinates for each vertex.

***format* (optional input parameter of type *addParameter*)** is a string to define the mesh type. Only possible values are 'freefem', 'gmsh', 'medit' or 'triangle'. Default value is 'freefem'. In 3D 'freefem' and 'medit' formats are supposed to be identical.

***Th* (output argument)** is a mesh structure. See 4.2.

In the following, the mesh structure Th is defined using *GetMeshOpt* function for 2D and 3D examples. The function *PrintMesh* gives statistics about the mesh (number of vertices or elements, mesh step size, ...)

4.4.4 Examples

Reading 2D *FreeFem++* mesh files

By default 2D mesh format is freefem (FreeFem++ format).

```
Th=GetMeshOpt('disk4-1-50.msh',2);
PrintMesh(Th);
```

Listing 1: 2D FreeFem++ mesh

```
---------------
Mesh: disk4-1-50.msh
dim=2, d=2, format=freefem
nq=3576, nme=6950, nbe=200
hmax=0.055886, hmin=0.022214
---------------
```

Reading 2D *Triangle* mesh files

For *Triangle* mesh files, the user has to set the format option as *triangle*.

```
Th=GetMeshOpt('box.1',2,'format','triangle');
PrintMesh(Th);
```

Listing 2: 2D Triangle mesh

```
---------------
Mesh: box.1
dim=2, d=2, format=triangle
nq=12, nme=12, nbe=12
hmax=1.500000, hmin=1.000000
---------------
```

Reading 2D or 3D *medit* mesh files

3D default mesh format is *medit*. Two examples for medit are given in the following, for each dimension

```
Th=GetMeshOpt('rect_bis2.mesh',2, ...
    'format','medit');
PrintMesh(Th);
```

Listing 3: 2D medit mesh

```
---------------
Mesh: rect_bis2.mesh
dim=2, d=2, format=medit
nq=2691, nme=5192, nbe=188
hmax=0.023424, hmin=0.009316
---------------
```

```
Th=GetMeshOpt('cube6-1-3.mesh',3);
PrintMesh(Th);
```

Listing 4: 3D medit mesh

```
---------------
Mesh: cube6-1-3.mesh
dim=3, d=3, format=medit
nq=64, nme=162, nbe=108
hmax=0.577351, hmin=0.333333
---------------
```

Reading 2D or 3D *gmsh* mesh files

For a gmsh file, the 'format' option with value 'gmsh' is required.

```
Th=GetMeshOpt('magnetism.msh',2, ...
    'format','gmsh');
PrintMesh(Th);
```

Listing 5: 2D gmsh mesh

```
---------------
Mesh: magnetism.msh
dim=2, d=2, format=gmsh
nq=1924, nme=3720, nbe=126
hmax=0.068956, hmin=0.022419
---------------
```

```
Th=GetMeshOpt('sphere8-4.msh',3, ...
    'format','gmsh');
PrintMesh(Th);
```

Listing 6: 3D gmsh mesh

```
---------------
Mesh: sphere8-4.msh
dim=3, d=3, format=gmsh
nq=1769, nme=7214, nbe=2022
hmax=0.322268, hmin=0.067119
---------------
```

5 Plotting a mesh

In this section we present some functions to represent the volume of a mesh and mesh boundaries. Some other functions are also presented and can be used for debugging as they provide vertex, element or boundary element numbers.

5.1 *PlotMesh* function

5.1.1 Description

PlotMesh function allows to display a mesh in 2D or 3D. The only required argument is the mesh structure.

5.1.2 Usage

- Basic usage

```
Th=GetMeshOpt(...);
PlotMesh(Th);
```

- With all options

```
Th=GetMeshOpt(...);
PlotMesh(Th,'LineWidth',...,'Color',...,'RGBcolors',..., ...
    'FaceAlpha',...,'labels',...);
```

5.1.3 Arguments

***Th* (input parameter)** is a mesh structure (see 4.2)

***LineWidth* (optional parameter of type *addParameter*)** is a double which sets the line width of mesh lines. Default value is 0.5.

***Color* (optional parameter of type *addParameter*)** is a string which defines the color of mesh lines. Default value is the empty string.

***RGBcolors* (optional parameter of type *addParameter*)** is an array of RGB values (doubles between 0 and 1) to set RGB values of the mesh lines. Each region may be identified by a different RGB. Default value is the empty array.

***FaceAlpha* (optional parameter of type *addParameter*)** is an integer which sets the transparency (only in 3D). 'FaceAlpha'=0 means no transparency. Default value is 0.

***labels* (optional parameter of type *addParameter*)** is an array of labels (integer) to plot only specific regions.

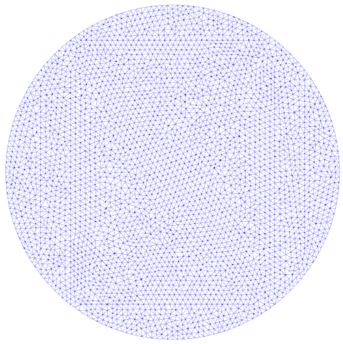
***Legend* (optional parameter of type *addParameter*)** is a boolean to display the legend or not. Default value is false. If value is true, the Ω symbol is used to denote each region.

No output argument

If both 'Color' and 'RGBcolors' are empty then RGBcolors prevails and is defined by the function *select_colors* of Timothy E. Holy - see [8] which allows to set a different color to each subdomain. If both 'Color' and 'RGBcolors' are unempty then 'Color' prevails. As 'Color' is given by a string, it is converted to an RGB value using the table given in Scalable Vector Graphics W3C web site.

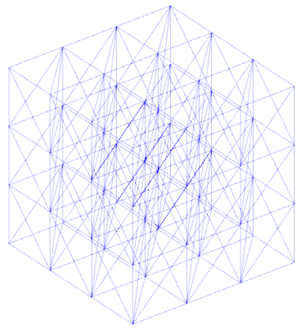
5.1.4 Examples

Listings 7 and 8 show basic examples of using *PlotMesh* in 2D and 3D for a FreeFem++ and a Medit mesh respectively. Listings 9 and 10 deal with the use of *LineWidth* and *RGBcolors* options. Using other options is similar. Listings 11 and 12 are examples of 2D and 3D meshes made of several physical domains.



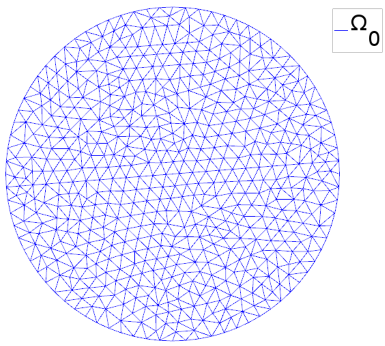
```
Th=GetMeshOpt('disk4-1-50.msh',2);
PlotMesh(Th);
```

Listing 7: 2D *PlotMesh* sample



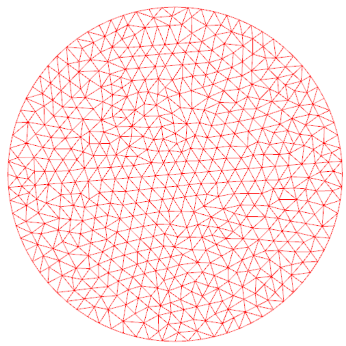
```
Th=GetMeshOpt('cube6-1-3.mesh',3);
PlotMesh(Th);
```

Listing 8: 3D *PlotMesh* sample



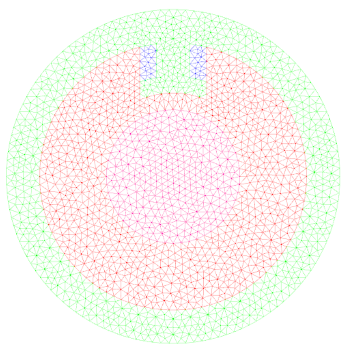

```
Th=GetMeshOpt('disque4-1-20.msh',2);
PlotMesh(Th,'LineWidth',1.0, ...
     'Legend',true);
```

Listing 9: 2D *PlotMesh* sample with a legend



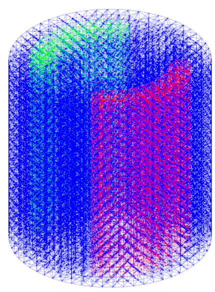
```
Th=GetMeshOpt('disque4-1-20.msh',2);
PlotMesh(Th,'LineWidth', 1.0, ...
     'RGBcolors', [1 0 0])
```

Listing 10: 2D *PlotMesh* sample with a color



```
Th=GetMeshOpt('magnetism.msh',2, ...
     'format','gmsh');
PlotMesh(Th);
```

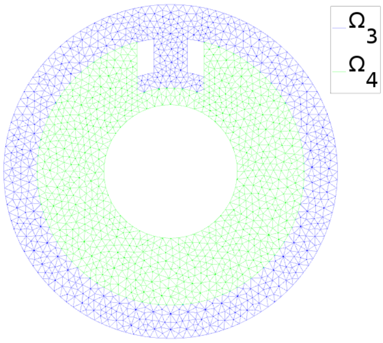
Listing 11: 2D *PlotMesh* sample



```
Th=GetMeshOpt( ...
     'FlowVelocity3d01-3.mesh',3);
PlotMesh(Th);
```

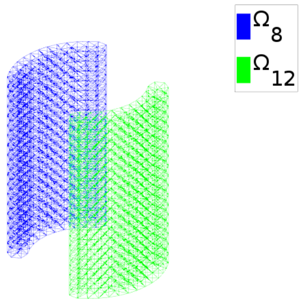
Listing 12: 3D *PlotMesh* sample

In Listing 14 only labels 8 and 12 are shown (to be compared with Listing 12).




```
Th=GetMeshOpt('magnetism.msh',2, ...
      'format','gmsh');
PlotMesh(Th,'labels',[3 ...
      4],'Legend',true);
```

Listing 13: 2D *PlotMesh* using *labels* option




```
Th=GetMeshOpt( ...
      'FlowVelocity3d01-3.mesh',3);
PlotMesh(Th,'labels',[8 ...
      12],'Legend',true);
```

Listing 14: 3D *PlotMesh* using *labels* option

5.2 *PlotBounds* function

5.2.1 Description

PlotBounds function allows to display the boundaries of a 2D or 3D mesh. The only required argument is the mesh structure.

5.2.2 Usage

- Basic usage

```
Th=GetMeshOpt(...);
PlotBounds(Th);
```

- With all options

```
Th=GetMeshOpt(...);
rgbcol=PlotBounds(Th,'LineWidth',...,'Color',..., ...
     'RGBcolors',...,'Legend',...,'FontSize',...,'labels',...);
```

5.2.3 Arguments

Th **(input parameter)** is a mesh structure (see 4.2)

LineWidth **(optional parameter of type *addParameter*)** is a double which sets the line width of mesh lines. Default value is 2.

Color **(optional parameter of type *addParameter*)** is a string which defines the color of mesh lines. Default value is the empty string.

RGBcolors **(optional parameter of type *addParameter*)** is an array of RGB values (doubles between 0 and 1) to set RGB values of the mesh lines. Each boundary may be identified by a different RGB. Default value is the empty array.

Legend **(optional parameter of type *addParameter*)** is a bool to display the legend or not. Default value is *true*. The Γ symbol is used to denote each boundary.

FontSize **(optional parameter of type *addParameter*)** is an integer to set the font size of the legend. Default value is 10.

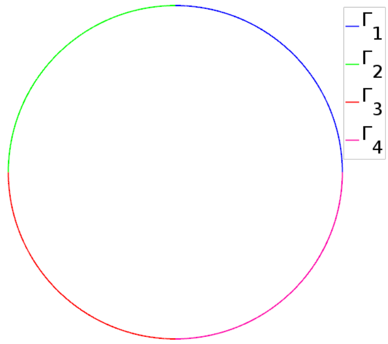
labels **(optional parameter of type *addParameter*)** is an array of labels (integer) to plot only specific regions.

rgbcol **optional output argument** is the array of RGB colors used by the plot.

If 'Color' and 'RGBcolors' are both empty, as for *PlotMesh* in 5.1 then RGBcolors prevails and is defined by the function *select_colors* of Timothy E. Holy - see [8] which allows to set a different color to each subdomain. If both 'Color' and 'RGBcolors' are unempty then Color prevails. If 'Color' is given by a string, it is converted to an RGB value using the table given in Scalable Vector Graphics W3C web ite

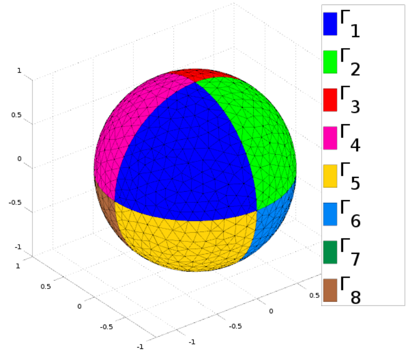
5.2.4 Examples

Listings 15 and 16 show basic examples of using *PlotBounds* in 2D and 3D. Listing 17 deals with the use of *Color* option. Using other options is similar. Listings 18 and 19 show examples combining *PlotMesh* and *PlotBounds* functions and also the use of *labels* option. On figures of Listings 18, 19, and 21 it is not possible to have legends for both domains and boundaries.



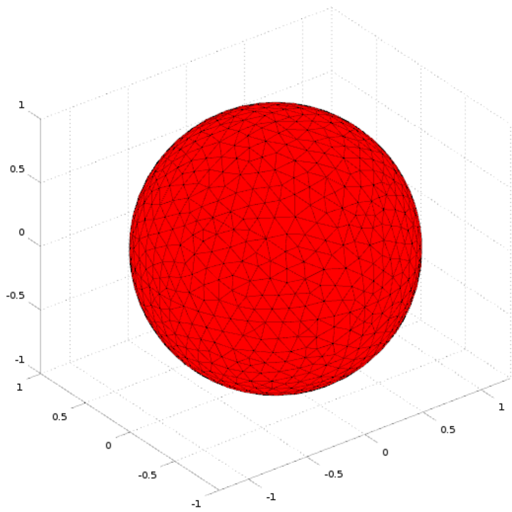

```
Th=GetMeshOpt('disk4-1-50.msh',2);
PlotBounds(Th);
```

Listing 15: 2D *PlotBounds* sample



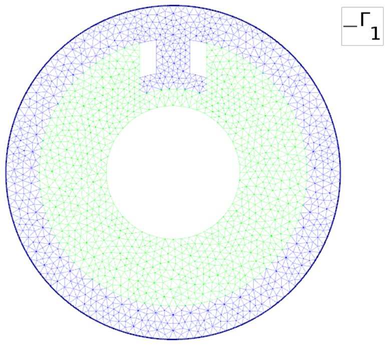

```
Th=GetMeshOpt('sphere8-4.msh',3, ...
    'format', 'gmsh');
PlotBounds(Th);
```

Listing 16: 3D *PlotBounds* sample



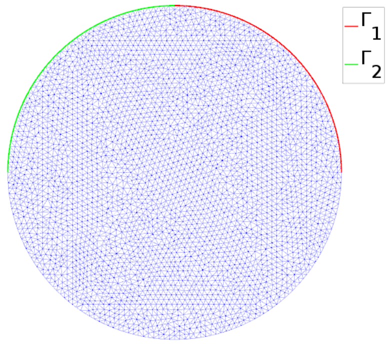

```
Th=GetMeshOpt('sphere8-4.msh',3, 'format', 'gmsh');
PlotBounds(Th,'Color','red','Legend',false);
```

Listing 17: 3D *PlotBounds* sample in only red color



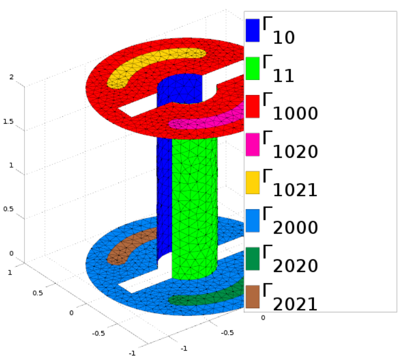

```
Th=GetMeshOpt('magnetism.msh',2, ...
    'format','gmsh');
PlotBounds(Th,'Color','black');
PlotMesh(Th,'labels',[3,4]);
```

Listing 18: *PlotMesh+PlotBounds* sample



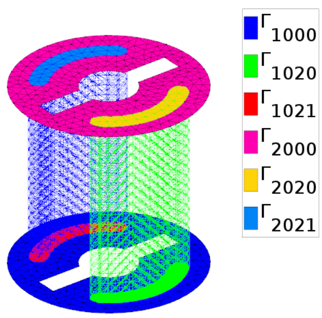

```
Th=GetMeshOpt('disk4-1-50.msh',2);
PlotMesh(Th);
PlotBounds(Th,'labels',[1,2], ...
    'RGBcolors',[1,0,0;0,1,0]);
```

Listing 19: *PlotBounds+ PlotMesh* sample




```
Th=GetMeshOpt('cylinderkey-10.msh', ...
    3, 'format','gmsh');
PlotBounds(Th,'labels',[10 11 1000 ...
    1020 1021 2000 2020 2021])
```

Listing 20: 3D *PlotBounds* sample




```
Th=GetMeshOpt( ...
    'FlowVelocity3d01-3.mesh',3);
PlotBounds(Th,'labels',[1000 1020 ...
    1021 2000 2020 2021]);
PlotMesh(Th,'labels',[8,12]);
```

Listing 21: *PlotBounds+ PlotMesh* sample

5.3 *PlotMeshNodeNumber* function

5.3.1 Description

PlotMeshNodeNumber function allows to display numbers of nodes in a mesh. The only required argument is the mesh structure. This function could be useful for debugging.

The function is limited to the 2D case.

5.3.2 Usage

- Basic usage

```
Th=GetMeshOpt(...);
PlotMeshNodeNumber(Th);
```

- With all options

```
Th=GetMeshOpt ( . . . ) ;
PlotMeshNodeNumber (Th, 'BackgroundColor' , . . . , 'FontSize' , . . . , 'Color' , . . . , 'C' , . . . ) ;
```

5.3.3 Arguments

***Th* (input parameter)** is a mesh structure (see 4.2)

***BackgroundColor* (optional parameter of type *addParameter*)** is a RGB value which sets the background color. Default value is [1 1 1] (white).

***FontSize* (optional parameter of type *addParameter*)** is an integer to set the font size of node numbers. Default value is 10.

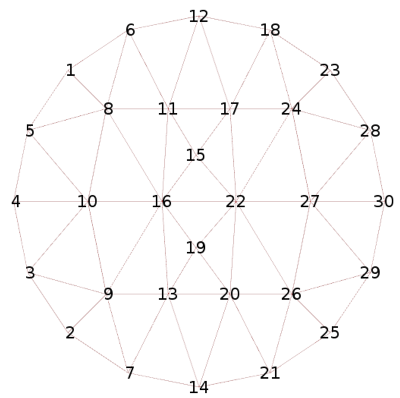
***Color* (optional parameter of type *addParameter*)** is a RGB value (doubles between 0 and 1) to set RGB value of the number color. Default value is [0 0 0] (black).

***C* (optional parameter of type *addParameter*)** is an integer equal to the shift for node numbering (equal to 0 or 1). Default value is 0. If C=0, node numbering goes from 1 to Th.nq else from 0 to Th.nq-1.

No output argument

5.3.4 Examples

Listing 22 is an example of using *PlotMeshNodeNumber* combined with *PlotMesh*.




```
Th=GetMeshOpt ( 'disque4 −1−4.msh' ,2) ;
PlotMesh (Th, 'Color' , 'brown' )
PlotMeshNodeNumber (Th, 'Color' ,[0 0 ...
      0] , 'FontSize' ,22) ;
```

Listing 22: *PlotMeshNodeNumber* sample

5.4 *PlotMeshTriangleNumber* function

5.4.1 Description

PlotMeshTriangleNumber function allows to display numbers of triangles in a 2D mesh. The only required argument is the mesh structure. This function could be useful for debugging. A call to the function *PlotMesh* is required before the call to *PlotMeshTriangleNumber*.

The function is limited to the 2D case.

5.4.2 Usage

- Basic usage

```
Th=GetMeshOpt(...);
PlotMeshTriangleNumber(Th);
```

- With all options

```
Th=GetMeshOpt(...);
PlotMeshTriangleNumber(Th,'BackgroundColor',...,'FontSize',..., ...
    'Color',...,  'EdgeColor,...,'C',...);
```

5.4.3 Arguments

***Th* (input parameter)** is a mesh structure (see 4.2)

***BackgroundColor* (optional parameter of type *addParameter*)** is a RGB value which sets the background color. Default value is [1 1 1] (white).

***FontSize* (optional parameter of type *addParameter*)** is an integer to set the font size of triangle numbers. Default value is 10.

***Color* (optional parameter of type *addParameter*)** is a RGB value (doubles between 0 and 1) to set RGB value of the number color. Default value is [0 0 0] (black).

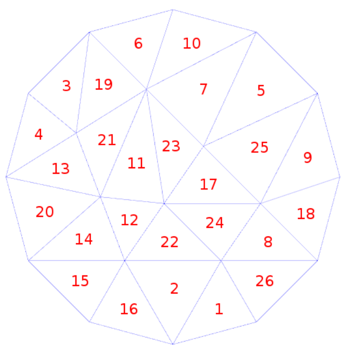
***EdgeColor* (optional parameter of type *addParameter*)** is a RGB value (doubles between 0 and 1) to set RGB value of the number color on edges. Default value is [0 0 0] (black).

***C* (optional parameter of type *addParameter*)** is an integer equal to 0 or 1 for the numbering shift (if C=1, numbering starts from 0). Default value is 0.

No output argument

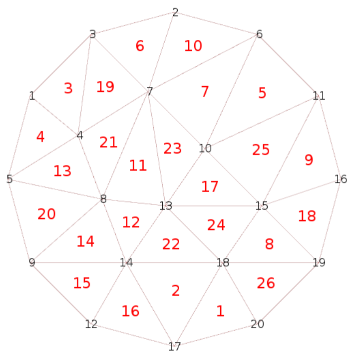
5.4.4 Examples

Listing 23 is an example of using *PlotMeshTriangle* combined with *PlotMesh*. Listing 24 depicts a more complete example including *PlotMesh*, *PlotMeshNodeNumber* and *PlotMeshTriangleNumber*.




```
Th=GetMeshOpt('disque4-1-3.msh',2);
PlotMesh(Th);
PlotMeshTriangleNumber(Th,'Color',[1 ...
    0 0],'FontSize',22);
```

Listing 23: *PlotMeshTriangleNumber* sample



```
Th=GetMeshOpt('disque4-1-3.msh',2);
PlotMesh(Th,'Color','brown');
PlotMeshNodeNumber(Th,'FontSize',16);
PlotMeshTriangleNumber(Th,'Color',[1 ...
    0 0],'FontSize',22);
```

Listing 24: *PlotMeshTriangleNumber*+ *PlotMeshNodeNumber* sample

5.5 *PlotBoundsEdgeNumber* function

5.5.1 Description

PlotBoundsEdgeNumber function allows to display numbers of boundary edges in a 2D mesh. The only required argument is the mesh structure. This function could be useful for debugging. A call to the function *PlotMesh* is required before the call to *PlotBoundsEdgeNumber*.

The function is limited to the 2D case.

5.5.2 Usage

- Basic usage

```
Th=GetMeshOpt(...);
PlotBoundsEdgeNumber(Th);
```

- With all options

```
Th=GetMeshOpt(...);
PlotBoundsEdgeNumber(Th,'RGBTextColors',...,  'RGBEdgeColors',..., ...
    'BackgroundColor',...,'FontSize',...,'FontWeight',...,'Color',..., ...
    'EdgeColor',...,'LineWidth',...,  'LineStyle',...);
```

5.5.3 Arguments

***Th* (input parameter)** is a mesh structure (see 4.2)

***RGBTextColors* (optional parameter of type *addParameter*)** sets the RGB color of boundary edge numbers. Default value is an empty array.

***RGBEdgeColors* (optional parameter of type *addParameter*)** sets the RGB color of boundary box edge numbers. Default value is an empty array.

***BackgroundColor* (optional parameter of type *addParameter*)** is a RGB value which sets the boundary edge number background box color. Default value is [1 1 1] (white).

***FontSize* (optional parameter of type *addParameter*)** is an integer to set the font size of boundary node numbers. Default value is 10.

***FontWeight* (optional parameter of type *addParameter*)** is a string to define set the boundary edge number font weight as 'normal', 'bold','light' or 'demi'. Default value is 'normal'.

***Color* (optional parameter of type *addParameter*)** is a RGB value (doubles between 0 and 1) to set RGB value of the number color. Default value is [0 0 0] (black).

***EdgeColor* (optional parameter of type *addParameter*)** is a RGB value (doubles between 0 and 1) to set RGB value of the number color on edges. Default value is [0 0 0] (black).

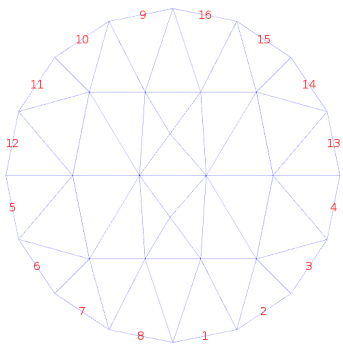
***LineWidth* (optional parameter of type *addParameter*)** is a double which sets the line width of mesh lines. Default value is 0.5.

***LineStyle* (optional parameter of type *addParameter*)** sets the line style of mesh lines to 'none', '-','--','.' or '-.' Default value is 'none'.

No output argument

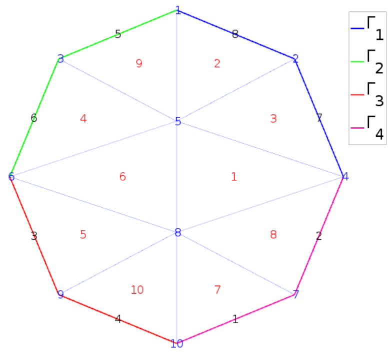
5.5.4 Examples

Listing 25 is an example of using *PlotBoundsEdge* combined with *PlotMesh*. Listing 26 depicts a more complete example including *PlotMesh*, *PlotMeshNodeNumber* and *PlotMeshTriangleNumber*.



```
Th=GetMeshOpt('disque4-1-4.msh',2);
PlotMesh(Th);
PlotBoundsEdgeNumber(Th,'Color',[1 ...
    0 0],'FontSize',16);
```

Listing 25: *PlotBoundsEdgeNumber* sample



```
Th=GetMeshOpt('disque4-1-2.msh',2);
PlotMesh(Th);
RGBcolors=PlotBounds(Th,'LineWidth',2);
PlotBoundsEdgeNumber(Th, ...
    'RGBEdgeColors', ...
    RGBcolors,'Color',[0 0 ...
    0],'LineStyle','-', ...
    'LineWidth',0.5,'FontSize',16);
PlotMeshNodeNumber(Th,'Color',[0 0 ...
    1],'FontSize',16);
PlotMeshTriangleNumber(Th,'Color',[1 ...
    0 0],'FontSize',16);
```

Listing 26: *PlotBoundsEdgeNumber*+ *PlotMeshNodeNumber*+ *PlotMeshTriangleumber* sample

5.6 *PlotBasisFunc* function

5.6.1 Description

PlotBasisFunc function allows to display the $\mathbb{P}^1$ basis function of a given vertex. The mesh structure and the vertex number are required. This function could be useful for debugging.

The function is limited to the 2D case.

5.6.2 Usage

- Basic usage

```
Th=GetMeshOpt(...);
n=...;
PlotBasisFunc(Th,n);
```

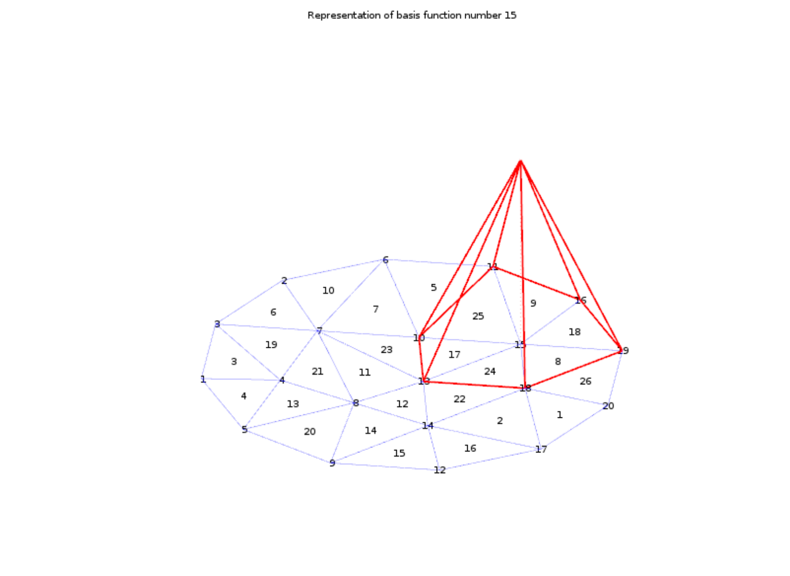
5.6.3 Arguments

***Th* (input parameter)** is a mesh structure (see 4.2)

***n* (input parameter)** vertex number for which the basis function is displayed

No output argument

5.6.4 Examples




```
Th=GetMeshOpt('disque4-1-3.msh',2);
n=15;PlotBasisFunc(Th,n);
```

Listing 27: 2D *PlotBasisFunc* sample

6 Representation of nodal variables

6.1 *PlotVal* function (2D)

6.1.1 Description

PlotVal function allows to display fields corresponding to nodal variables in 2D. Required arguments are the mesh structure and the field u. u is an array of doubles of size n_q.

6.1.2 Usage

- Basic usage

```
Th=GetMeshOpt(...);
u=...;
PlotVal(Th,u);
```

- With all options

```
Th=GetMeshOpt(...);
u=...;
h=PlotVal(Th,u,'CameraPosition',...,'colormap',...,  ...
     'shading',...,'colorbar',...,'caxis',...,  'labels',...);
```

6.1.3 Arguments

***Th* (input parameter)** is a mesh structure (see 4.2)

***Val* (input parameter)** is an array of doubles of size Th.nq

***CameraPosition* (optional parameter of type *addParameter*)** is an array of two doubles to set the camera position. Default value is *view(2);*.

***colormap* (optional parameter of type *addParameter*)** is a colormap value. Default value is 'jet'.

shading **(optional parameter of type *addParameter*)** is a bool to use 'shading interp' or not. Default value is *true*.

colorbar **(optional parameter of type *addParameter*)** is a bool to display the colorbar or not. Default value is *true*.

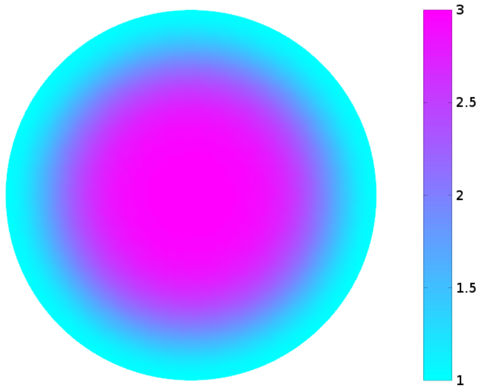
caxis **(optional parameter of type *addParameter*)** is an array of four doubles to set the axis. Default value is an empty array.

labels **(optional parameter of type *addParameter*)** is an array of labels (integer) to plot only specific regions.

h **optional output argument** returns the handle to the figure

6.1.4 Examples

Listing 28 shows an example of *PlotVal* option. Using other options is similar.




```
Th=GetMeshOpt('disk4-1-50.msh',2);
x=Th.q(1,:);y=Th.q(2,:);
u=2+cos(pi*(x.^2+y.^2));
PlotVal(Th,u,'colormap','cool');
```

Listing 28: 2D *PlotVal* sample

6.2 *Plot3DSurfVal* function

6.2.1 Description

Plot3DSurfVal allows to represent the variables on the surface of a 3D mesh. *Plot3DSurfVal* function allows to display fields corresponding to nodal variables on the surface of a 3D mesh. Required arguments are the mesh structure and the field u. u is an array of doubles of size n_q.

6.2.2 Usage

- Basic usage

```
Th=GetMeshOpt(...);
u=...;
Plot3DSurfVal(...);
```

- With all options

```
Th=GetMeshOpt(...);
u=...;
Plot3DSurfVal(Th,u,'colormap',...,'colorbar',..., ...
    'labels',...,'PlotOptions',...);
```

6.2.3 Arguments

***Th* (input parameter)** is a mesh structure (see 4.2)

***u* (input parameter)** array of doubles of size Th.nq

***colormap* (optional parameter of type *addParameter*)** is a colormap value. Default value is 'jet'.

***colorbar* (optional parameter of type *addParameter*)** is a bool to display the colorbar or not. Default value is *true*.

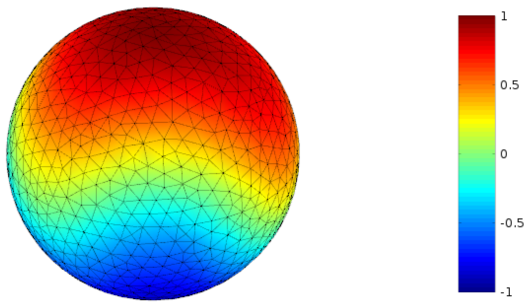
***labels* (optional parameter of type *addParameter*)** is an array of labels (integer) to plot only on specific boundaries.

***PlotOptions* (optional parameter of type *addParameter*)** is a cell for defining plotting options. Default value is an empty cell.

No output argument

6.2.4 Examples

For example




```
Th=GetMeshOpt('sphere8-4.msh',3,'format','gmsh');
f=@(x,y,z) x.*y.^2+z;
Plot3DSurfVal(Th,f(Th.q(1,:),Th.q(2,:),Th.q(3,:)));
```

Listing 29: 3D *Plot3DSurfVal* sample

6.3 *PlotIsolines* function (2D)

6.3.1 Description

PlotIsolines function allows to display isolines of a scalar field in 2D. Required arguments are the mesh structure and the field U. U is an array of doubles of size n_q. The function uses the external toolboxes *gptoolbox* (see [9]) and *colorbarf* ([6]).

6.3.2 Usage

- Basic usage

```
Th=GetMeshOpt(...);
U=...;
PlotIsolines(Th,U);
```

- With all options

```
Th=GetMeshOpt(...);
U=...;
[col,isor]=PlotIsolines(Th,U,'niso',...,'colormap',...,'colorbar',..., ...
    'PlotOptions',...,'isorange',...,'labels',...);
```

6.3.3 Arguments

***Th* (input parameter)** is a mesh structure (see 4.2)

***U* (input parameter)** array of doubles of size Th.nq

***niso* (optional parameter of type *addParameter*)** is an integer which sets the number of isolines to be plotted. Default value is 10.

***colormap* (optional parameter of type *addParameter*)** is a colormap value. Default value is 'jet'.

***colorbar* (optional parameter of type *addParameter*)** is a bool to display the colorbar or not. Default value is *false*.

***PlotOptions* (optional parameter of type *addParameter*)** is a cell for defining plotting options. Default value is an empty cell.

***plan* (optional parameter of type *addParameter*)** (bool) To plot isolines in 2D in the xOy-plane (if true) or in 3D (if false).

***isorange* (optional parameter of type *addParameter*)** is an array of integers which defines the isoline values. Default value is an empty array and niso isoline values are set in proportion in the range of minimum and maximum values.

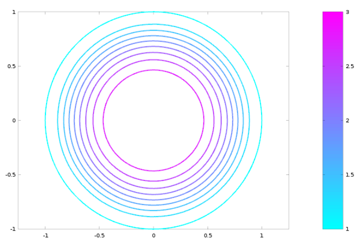
***labels* (optional parameter of type *addParameter*)** is an array of labels (integer) to plot only specific regions. Default value is an empty array.

***[col,isor]* optional output arguments** . *col* is the array of RGB colors used by the plot. *isor* is the array of isoline values.

PlotIsolines allows to plot isolines for a 2D variable.

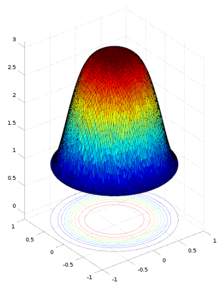
6.3.4 Examples

For example



```
Th=GetMeshOpt('disk4-1-50.msh',2);
x=Th.q(1,:);y=Th.q(2,:);
u=2+cos(pi*(x.^2+y.^2));
[colors,values]=PlotIsolines( ...
    Th,u, ...
    'colorbar',true,'PlotOptions', ...
    {'linewidth',2}, ...
    'colormap','cool');
```

Listing 30: *PlotIsolines* sample



```
Th=GetMeshOpt('disk4-1-50.msh',2);
x=Th.q(1,:);y=Th.q(2,:);
u=2+cos(pi*(x.^2+y.^2));
trisurf(Th.me',Th.q(1,:),Th.q(2,:),u);
PlotIsolines(Th,u,'plan',true);
if isOctave();axis([-1 1 -1 1 -0.1 ...
    3.1]);end
```

Listing 31: *PlotIsolines* sample

6.4 *PlotVal3D* function

Not yet implemented in Octave!!

6.5 *Plot3DSurfIsolines* function

6.5.1 Description

Plot3DSurfIsolines function allows to display isolines of a scalar field on the surface of a 3D mesh. Required arguments are the mesh structure and the field Val. Val is an array of doubles of size n_q.

6.5.2 Usage

- Basic usage

```
Th=GetMeshOpt(...);
Val=...;
Plot3DSurfIsolines(Th,Val);
```

- With all options

```
Th=GetMeshOpt(...);
Val=...;
Plot3DSurfIsolines(Th,Val,'niso',...,'colormap',...,  'colorbar',..., ...
    'PlotOptions',...,'isorange',...,'labels',...);
```

6.5.3 Arguments

***Th* (input parameter)** is a mesh structure (see 4.2)

***Val* (input parameter)** array of doubles of size Th.nq

***niso* (optional parameter of type *addParameter*)** is an integer which sets the number of isolines to be plotted. Default value is 10.

***colormap* (optional parameter of type *addParameter*)** is a colormap value. Default value is 'jet'.

***colorbar* (optional parameter of type *addParameter*)** is a bool to display the colorbar or not. Default value is *false*.

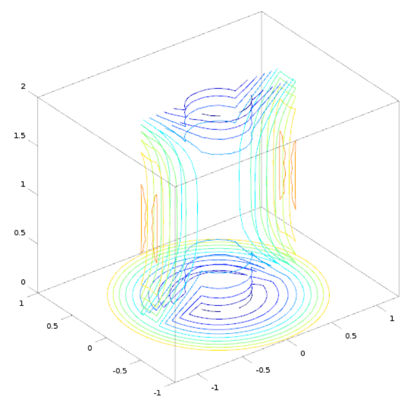
***PlotOptions* (optional parameter of type *addParameter*)** is a cell for defining plotting options. Default value is an empty cell.

***isorange* (optional parameter of type *addParameter*)** is an array of integers which defines the range of isoline values. Default value is an empty array and niso isoline values are set in proportion in the range of minimum and maximum values.

***labels* (optional parameter of type *addParameter*)** is an array of labels (integer) to plot only on specific boundaries. Default value is an empty array.

***[col,isor]* optional output arguments** . *col* is the array of RGB colors used by the plot. *isor* is the array of isoline values.

6.5.4 Examples



```
Th=GetMeshOpt('cylinderkey-10.msh',3, ...
     'format','gmsh');
f=@(x,y,z) x.^2+y.^2+cos((z-1).^2);
U=f(Th.q(1,:),Th.q(2,:),Th.q(3,:));
Plot3DSurfIsolines(Th,U,'labels', ...
    [10,11,31,2000,2020,2021],'niso', ...
    15,'PlotOptions', {'LineWidth',1})
view(3);
```

Listing 32: *Plot3DSurfIsolines* sample

7 Creating VTK files with *vtkWrite* function

There are several visualization tools which read VTK files: VisIt, Mayavi, ParaView, ... ParaView is used in the next figures.

The function *vtkWrite* in the package makes the conversion to the *VTK* format. This function not only writes the mesh structure into a file but also scalar or vector nodal values.

The *VTK* file can be loaded by ParaView. To really proceed the visualization, the user has to know how to use ParaView.

7.1 Mesh representation in VTK format

7.1.1 Description

In the toolbox the function *vtkWrite* enables to create a VTK file containing the mesh properties. A VTK file name and a mesh structure are required.

7.1.2 Usage

- Basic usage

```
Th=GetMeshOpt(...);
vtkWrite(cFileName,Th);
```

7.1.3 Arguments

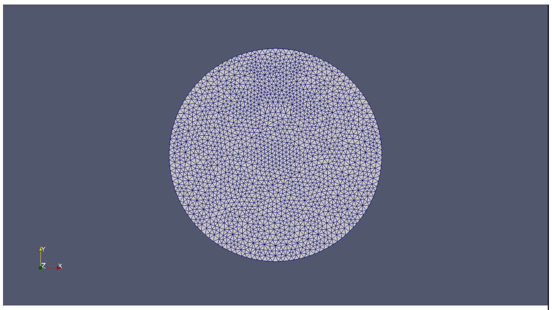
***cFileName* (input parameter)** is a string for the VTK file name (with .vtk extension)

***Th* (input parameter)** is a mesh structure (see 4.2)

No output argument

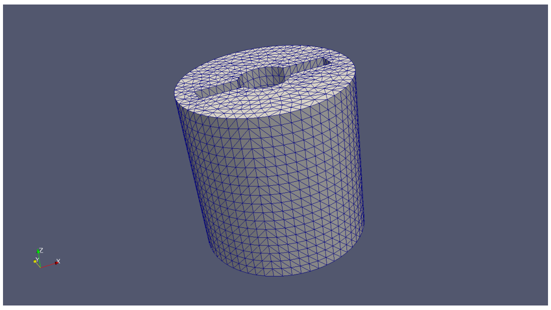
7.1.4 Examples

Listings 33 and 34 show examples of creating VTK files to represent a 2D or 3D mesh. The figures are identical to Listings 11 and 12.



```
Th=GetMeshOpt('magnetism.msh',2, ...
      'format','gmsh');
vtkWrite('magnetism.vtk',Th);
```

Listing 33: VTK file for a 2D mesh in ParaView



```
Th3=GetMeshOpt( ...
      'FlowVelocity3d01-3.mesh',3);
vtkWrite('flowvel3d.vtk',Th3);
```

Listing 34: VTK file for a 3D mesh in ParaView

7.2 Display of scalar results in VTK format

7.2.1 Description

The function *vtkWrite* also enables to create a VTK file containing the mesh properties and values to be displayed. A VTK file name, a mesh structure, a field and a string for the field name are required.

7.2.2 Usage

- Basic usage

```
Th=GetMeshOpt(...);
U=...;
names=...;
vtkWrite(cFileName,Th,U,names);
```

7.2.3 Arguments

***cFileName* (input parameter)** is a string which contains the VTK file name (with .vtk extension)

***Th* (input parameter)** is a mesh structure (see 4.2)

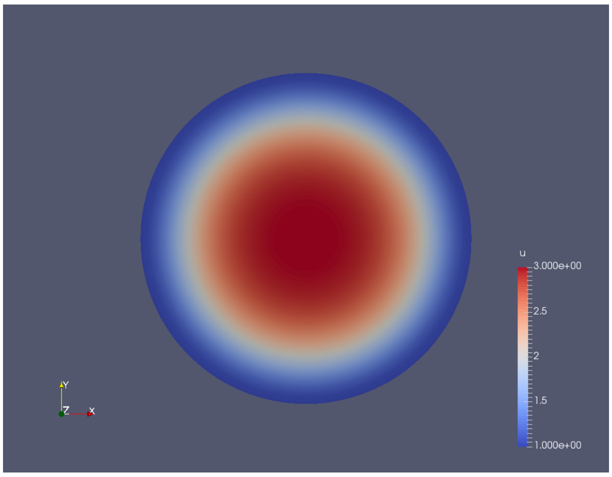
***U* (input parameter)** is a cell which contains all the field values to be saved in the VTK file (cell of dimension $\mathrm{Th.n_q}$)

***names* (input parameter)** is a cell variable which contains the names of the field values

No output argument

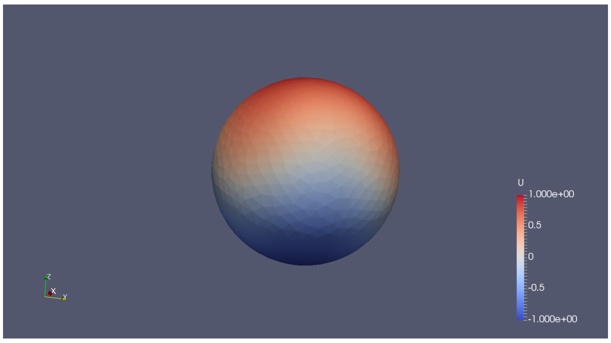
7.2.4 Examples

Listings 35 and 36 are examples of creation of VTK file containing mesh properties and scalar values. There are two scalar values in the 3D example.




```
Th=GetMeshOpt('magnetism.msh',2, ...
     'format','gmsh');
U=2+cos(pi*(Th.q(1,:).^2+Th.q(2,:).^2));
vtkWrite('magnetism-s.vtk',Th, ...
     {U'},{'u'});
```

Listing 35: VTK/2D scalar results




```
Th3=GetMeshOpt('sphere8-4.msh',3, ...
     'format','gmsh');
U=cos(Th3.q(1,:).^2+ ...
     Th3.q(2,:).^2+Th3.q(3,:).^2);
V=cos(Th3.q(1,:).^2+ ...
     Th3.q(2,:).^2+Th3.q(3,:).^2);
vtkWrite('sphere8-4-s.vtk',Th3, ...
     {U',V'},{'u','v'});
```

Listing 36: VTK/3D scalar results

7.3 Display of vector results in VTK format

7.3.1 Description

The function *vtkWrite* enables to create a VTK file containing the mesh properties and values to be displayed. A VTK file name, a mesh structure, a field and strings for the field name are required.

7.3.2 Usage

- Basic usage

7.3.3 Arguments

cFileName **(input parameter)** is a string which contains the VTK file name (with .vtk extension)

Th **(input parameter)** is a mesh structure (see 4.2)

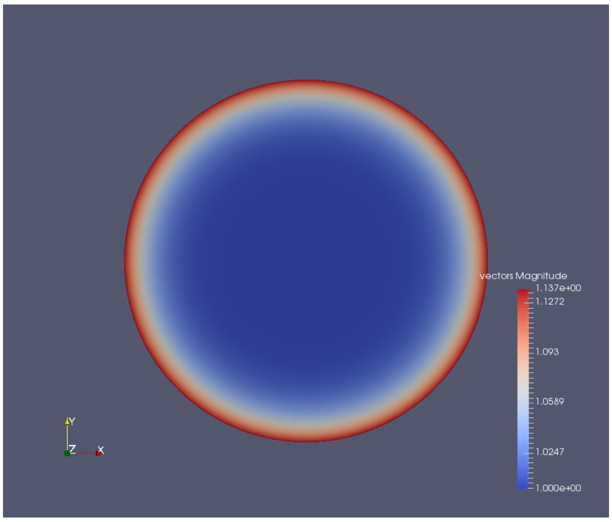
U **(input parameter)** is a cell which contains all the field values to be saved in the VTK file (cell of dimension $Th.n_q$)

names **(input parameter)** is a cell variable which contains the names of the field values

No output argument

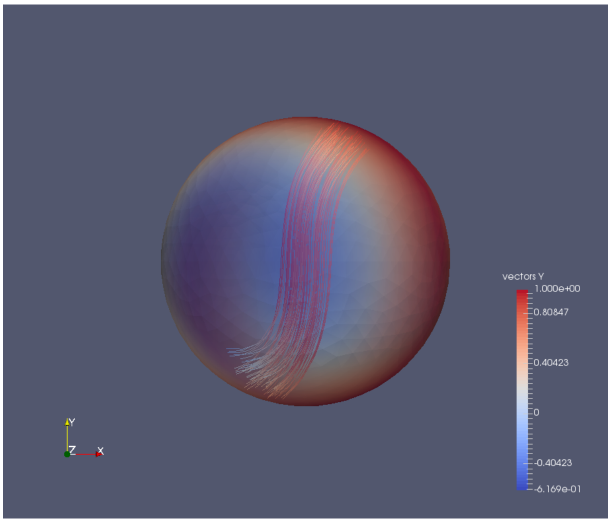
7.3.4 Examples

Listing 37 shows an example of VTK file showing the magnitude of a vector variable. Listing 38 is an example of VTK file showing the Y-component of a vector variable with streamlines.




```
Th=GetMeshOpt('magnetism.msh',2, ...
      'format','gmsh');
U{1}=Th.q(1,:).^2+Th.q(2,:).^2;
U{2}=cos(U{1});
vtkWrite('magnetism-v.vtk',Th, ...
      {[U{1}',U{2}']},{'U'});
```

Listing 37: vtk/2D vector results




```
Th3=GetMeshOpt('sphere8-4.msh',3, ...
      'format','gmsh');
U{1}=Th3.q(1,:).^2+Th3.q(2,:).^2+ ...
      Th3.q(3,:).^2;
U{2}=cos(Th3.q(1,:)-2*Th3.q(3,:));
U{3}=sin(Th3.q(1,:)+Th3.q(2,:));
vtkWrite('sphere8-4-v.vtk',Th3, ...
      {[U{1}',U{2}',U{3}']},{'U'});
```

Listing 38: vtk/3D vector results - streamlines

References

[1] F. Cuvelier, C. Japhet, and G. Scarella. OptFEM packages. http://www.math.univ-paris13.fr/~cuvelier/software, 2015.

[2] F. Cuvelier and G. Scarella. A generic way to solve partial differential equations by the $\mathbb{P}_1$-Lagrange finite element method in vector languages. https://www.math.univ-paris13.fr/~cuvelier/software/docs/Recherch/VecFEM/distrib/0.1b1/vecFEMP1_report-0.1b1.pdf, 2015.

[3] F. Cuvelier and G. Scarella. mVecFEMP1: a Matlab/Octave toolbox to solve boundary value and eigenvalues problems by a P1-Lagrange finite element method in any space dimension. http://www.math.univ-paris13.fr/~cuvelier/software/, 2015.

[4] P. J. Frey. Medit: An interactive mesh visualization software. https://www.ljll.math.upmc.fr/frey/publications/RT-0253.pdf, 2001.

[5] C. Geuzaine and J.-F. Remacle. Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009.

[6] B. Greenan. colorbarf: Add an accurate colorbar to your filled contour plot. http://www.mathworks.com/matlabcentral/fileexchange/1135-colorbarf/content/colorbarf.m, 2001.

[7] F. Hecht. New development in freefem++. *J. Numer. Math.*, 20(3-4):251–265, 2012.

[8] T. Holy. Generate maximally perceptually-distinct colors. http://www.mathworks.com/matlabcentral/mlc-downloads/downloads/submissions/29702/versions/3/download/zip, 2011.

[9] A. Jacobson et al. gptoolbox: Geometry Processing Toolbox. http://github.com/alecjacobson/gptoolbox, 2015.

[10] J. R. Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In Ming C. Lin and Dinesh Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, 1996.

Warning GIT! Working tree is dirty!!
GIT commit 626edb9c96ea05acc40407b7dfc32d3bbcdc7ecc
Date: Tue Nov 29 07:13:00 2016 +0100