



User's Guide*

François Cuvelier†

November 17, 2017

Abstract

This Octave package uses a `siMESH` object, coming from the `FC-SIMESH` package, to display simplicial meshes or datas on simplicial meshes. Its kernel uses the `FC-GRAPHICS4MESH` package.

Contents

1	Introduction	2
2	Installation	2
2.1	Installation automatic, all in one	2
3	Mesh	3
4	PLOTMESH function	4
5	PLOT function	9
6	PLOTISO function	13
7	SLICEMESH function	17
8	SLICE function	18
9	SLICEISO function	19

*Compiled with Octave 4.2.1

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

This work was supported by the ANR project DEDALES under grant ANR-14-CE23-0005.

1 Introduction

This **experimental** Octave package uses the Simplicial meshes could be:

- a triangular mesh in dimension 2, made with 2-simplices (ie. triangles),
- a tetrahedral mesh in dimension 3, made with 3-simplices (ie. tetrahedron),
- a triangular mesh in dimension 3 (surface mesh), made with 2-simplices,
- a line mesh in dimension 2 or 3 made with 1-simplices (ie. lines).

A simplicial mesh is given by its vertices array `q` and its connectivity array `me`. For demonstration purpose, some simplicial meshes are given in this package. They can be load by using the function `getMesh2D`, `getMesh3D` or `getMesh3Ds` of the `fc_graphics4mesh` package.

This package was tested under

Windows 10: with Octave 4.2.0 and 4.2.1

MacOS Sierra: with Octave 4.2.1 (install with homebrew)

Ubuntu 14.04.5 LTS: with Octave 4.2.1 and 4.2.0 (compiled from source, with graphic toolkit 'qt')

Ubuntu 16.04 LTS: with Octave 4.2.1 and 4.2.0 (compiled from source)

It is not compatible with Octave 4.0.x and previous.

2 Installation

2.1 Installation automatic, all in one

For this method, one just have to get/download the install file

`ofc_graphics4mesh_install.m`

or get it on the dedicated web page. Thereafter, one run it under Octave. This command download, extract and configure the `fc-graphics4mesh` and the required `fc-tools` package in the current directory.

For example, to install this package in directory `~/Octave/packages`, in a terminal one can do:

```
# mkdir -p ~/Octave/packages
# cd ~/Octave/packages
# wget ...
  http://www.math.univ-paris13.fr/~cuvelier/software/codes/fc-graphics4mesh/0.0.1/ofc_graphics4mesh.in
```

Then in a Octave terminal run the following commands

```
>> cd ~/Octave/toolboxes
>> ofc_graphics4mesh_install
```

There is the output of the `ofc_graphics4mesh_install` command:

```
1- Downloading and extracting the packages
  -> <fc-tools>[0.0.18] ... OK
  -> <fc-graphics4mesh>[0.0.1] ... OK
2- Setting the <fc-graphics4mesh> package
Write in ...
  ~/Octave/packages/fc-graphics4mesh-full/fc_graphics4mesh-0.0.1/configure_loc.m ...
  ...
  -> done
3- Using the <fc-graphics4mesh> package
Under Octave:
  addpath('~'/Octave/packages/./fc-graphics4mesh-full/fc_graphics4mesh-0.0.1')
  fc_graphics4mesh_init()

See ~/Octave/packages/ofc_graphics4mesh_set.m
```

The complete package (i.e. with all the other needed packages) is stored in the directory `~/Octave/packages/fc-graphics4mesh-full` and, for each Octave session, one have to set the package by:

```
>> ...
  addpath('~'/Octave/packages/fc-graphics4mesh-full/ofc-graphics4mesh-0.0.4')
>> fc_graphics4mesh_init()
```

For **uninstalling**, one just have to delete directory `~/Octave/packages/fc-graphics4mesh-full`

3 Mesh

The functions `getMesh2D`, `getMesh3D` and `getMesh3Ds` return a mesh vertices array q , a mesh elements connectivity array associated with the input argument d (simplex dimension) and the indices array `toGlobal`. The vertices array q is a dim -by- n_q array where dim is the space dimension (2 or 3) and n_q the number of vertices. The connectivity array me is a $(d + 1)$ -by- n_{me} array where n_{me} is the number of mesh elements and $0 \leq d \leq dim$ is the simplicial dimension:

- $d = 0$: points,
- $d = 1$: lines,
- $d = 2$: triangle,
- $d = 3$: tetrahedron.

So we can use theses functions to obtain

- 3D mesh: `getMesh3D(3)` (*main* mesh), `getMesh3D(2)`, `getMesh3D(1)`, `getMesh3D(0)`,
- 3D surface mesh: `getMesh3Ds(2)` (*main* mesh), `getMesh3Ds(1)`, `getMesh3Ds(0)`,
- 2D mesh: `getMesh2D(2)` (*main* mesh), `getMesh2D(1)`, `getMesh2D(0)`.

For example,

- `[q3,me3,toGlobal3]=fc_graphics4mesh.getMesh3D(3)` return a 3-simplicial mesh (main mesh) in space dimension $dim = 3$,
- `[q2,me2,toGlobal2]=fc_graphics4mesh.getMesh3D(2)` return a 2-simplicial mesh in space dimension $dim = 3$.

The third output are indices of the vertices in the *main* mesh:
`q3(:,toGlobal2) == q2`

4 PLOTMESH function

The method **PLOTMESH** displays the mesh or parts of the mesh defined by an **SIMESH** object.

Syntaxe

```
fc_siplt.plotmesh(Th,)  
fc_siplt.plotmesh(Th,Name,Value, ...)
```

Description

`fc_siplt.plotmesh(Th,)` displays all the $Th.d$ -dimensional simplices elements.

`fc_siplt.plotmesh(Th,Name,Value, ...)` specifies function options using one or more `Name,Value` pair arguments. Options of first level are

- `'d'` : to specify the dimension of the simplices elements (default : $Th.d$)
- `'labels'` : to select the labels of the elements to display,
- `'color'` : to specify the color of the displayed mesh elements. (default : use one color by displayed mesh elements),
- `'inlegend'` : add a legend name to graph if true (default : false)
- `'bounds'` : If true, draw the borders of the selected elementaries mesh elements (only for 2-dimensional simplices). (default : false)
- `'cutPlan'` : cut mesh by n plans given by n -by-4 array P where the equation of the i -th cut plan is given by

$$P(i,1)x + P(i,2)y + P(i,3)z + P(i,4) = 0.$$

The normal vector $P(i,1 : 3)$ pointed to the part of the mesh not displayed. (only for simplices in dimension 3) default : `[]` (no cut).

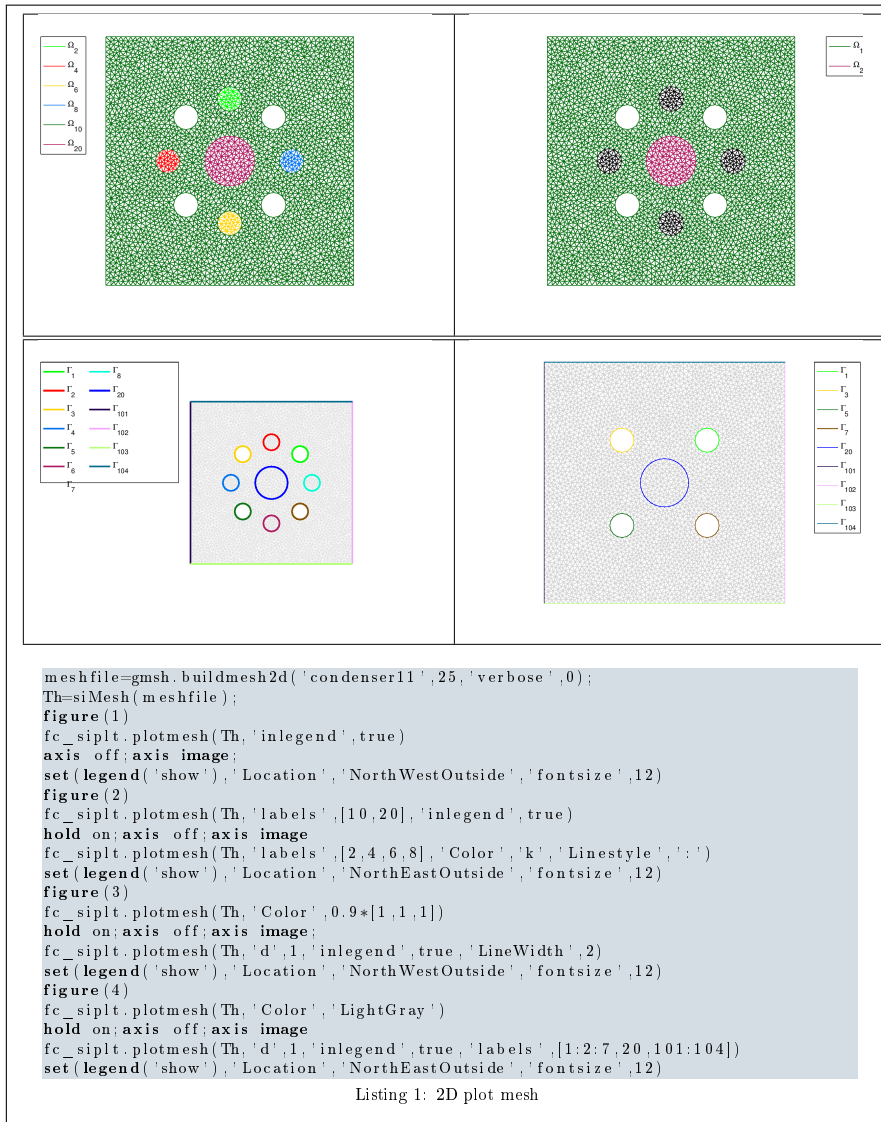
The options of second level depend on the type of elementaries mesh elements to represent.

One can use any option of the following functions according to the type of d -simplex to be represented.

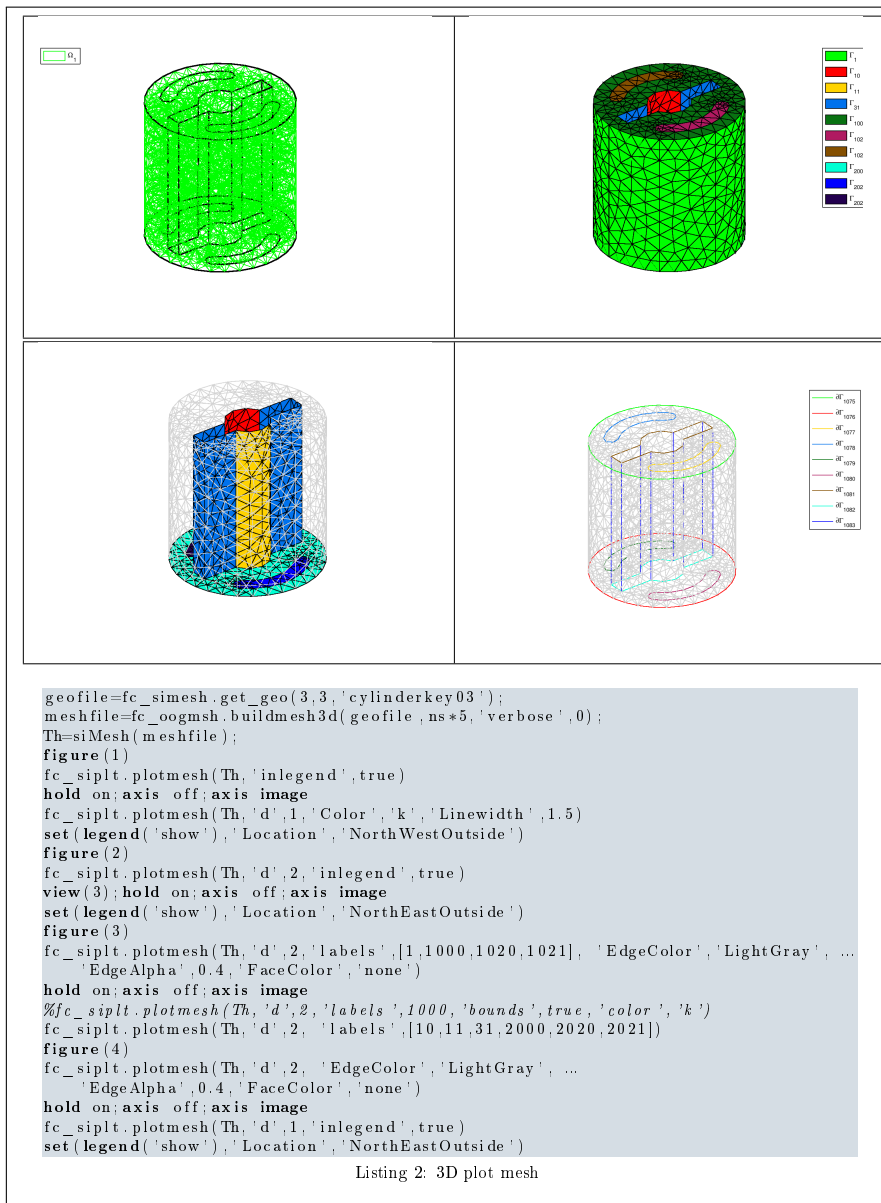
- In dimension 3,

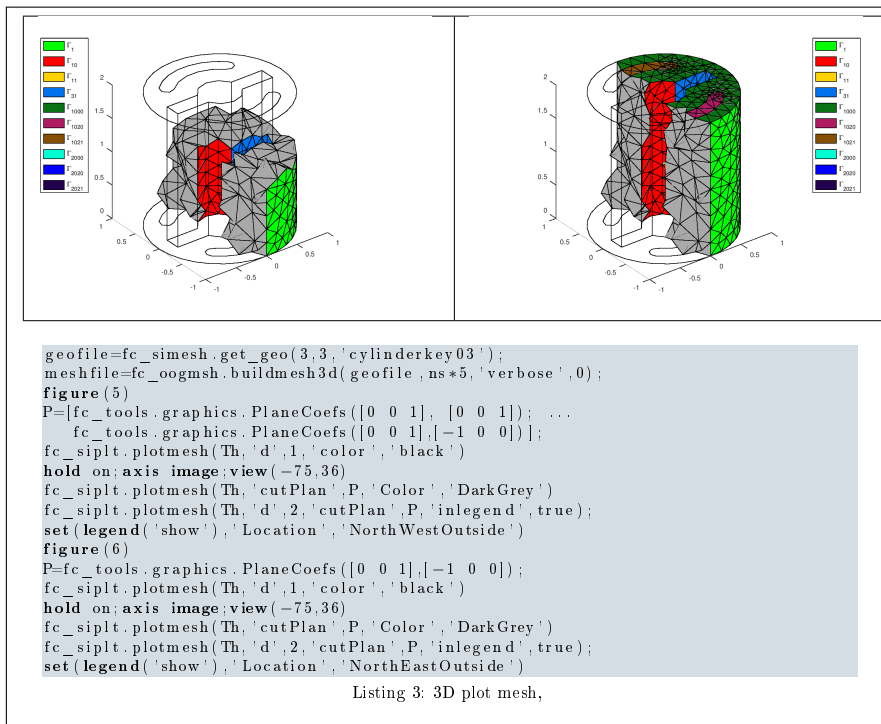
- if $d == 3$, **patch** function is used,
- if $d == 2$, **trimesh** function is used,
- if $d == 1$, **plot3** function is used,
- if $d == 0$, **plot3** function is used,
- In dimension 2,
 - if $d == 2$, **trimesh** function is used,
 - if $d == 1$, **plot** function is used,
 - if $d == 0$, **plot** function is used,
- In dimension 1,
 - if $d == 1$, **line** function is used,
 - if $d == 0$, **plot** function is used,

2D example The following example use the *.geo* file `condenser11.geo` which is in the directory `geodir` of the toolbox ...

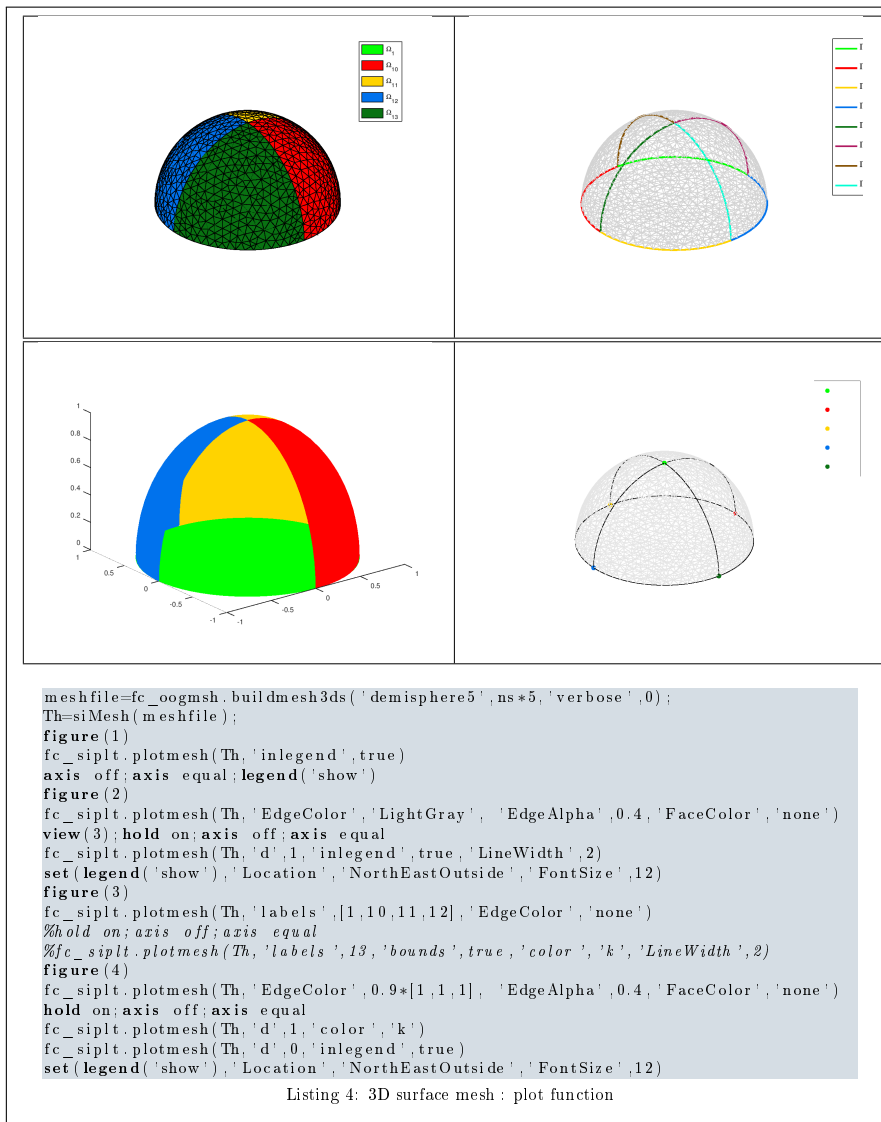


3D example The following example use the *.geo* file `cylinderkey.geo` which is in the directory `geodir` of the toolbox. This file contains description of a 3D mesh with simplices of dimensions 1, 2 and 3.





3D surface example The following example use the *.geo* file *demisphere5.geo* which is in the directory *geodir* of the toolbox. This file contains description of a 3D surface mesh with simplices of dimensions 1 and 2.



5 PLOT function

The method `FC_SIPLT.PLOT` displays scalar datas on the mesh or parts of the mesh defined by an `SI_MESH` object.

Syntaxe

```

fc_siplt.plot(Th,u)
fc_siplt.plot(Th,u,Name,Value,...)

```

Description

`fc_siplt.plot(Th,u)` displays data `u` on all the `Th.d`-dimensional simplices elements. The data `u` is an 1D-array of size `Th.nq` or `Th.nqGlobal` or `Th.nqParent`.

`fc_siplt.plot(Th,u,Name,Value, ...)` specifies function options using one or more `Name,Value` pair arguments. Options of first level are

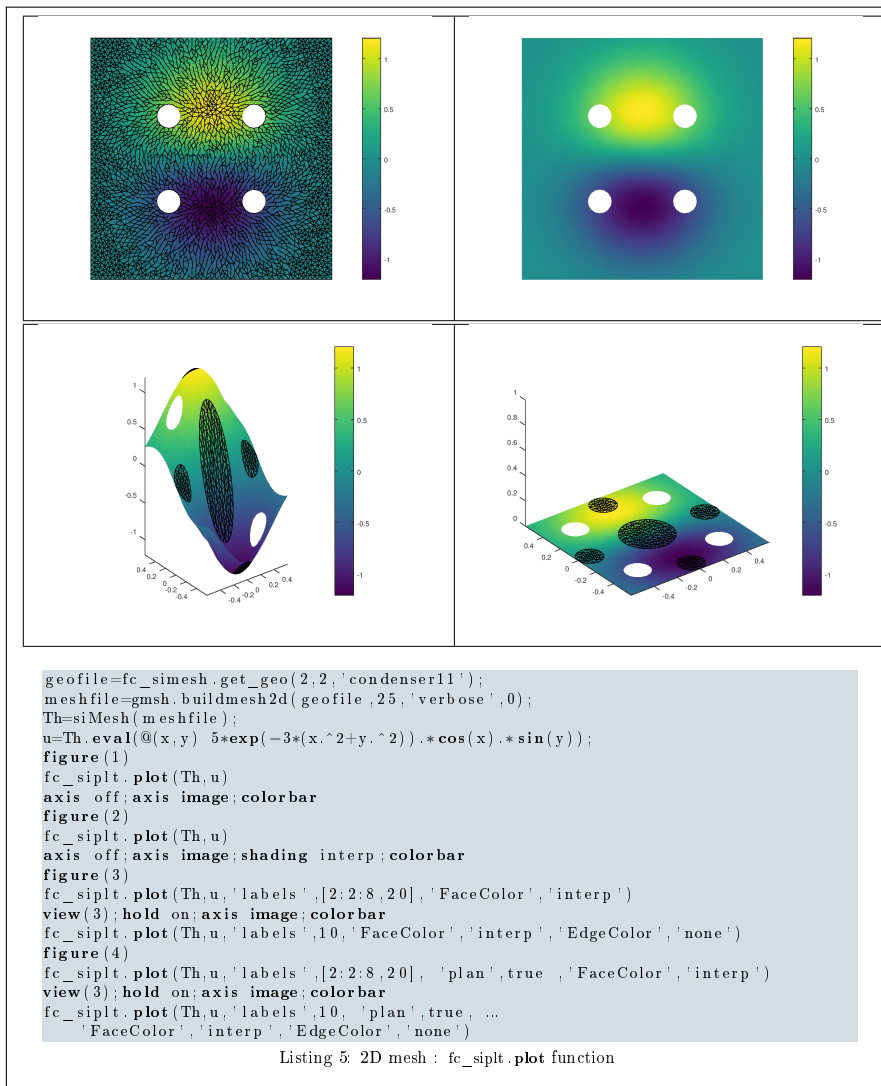
- `'d'` : to specify the dimension of the simplices elements (default : `Th.d`)
- `'labels'` : to select the labels of the elements to display data,
- `'plan'` : if true, (default : false)

The options of second level depend on the type of elementaries mesh elements on which we want to represent datas.

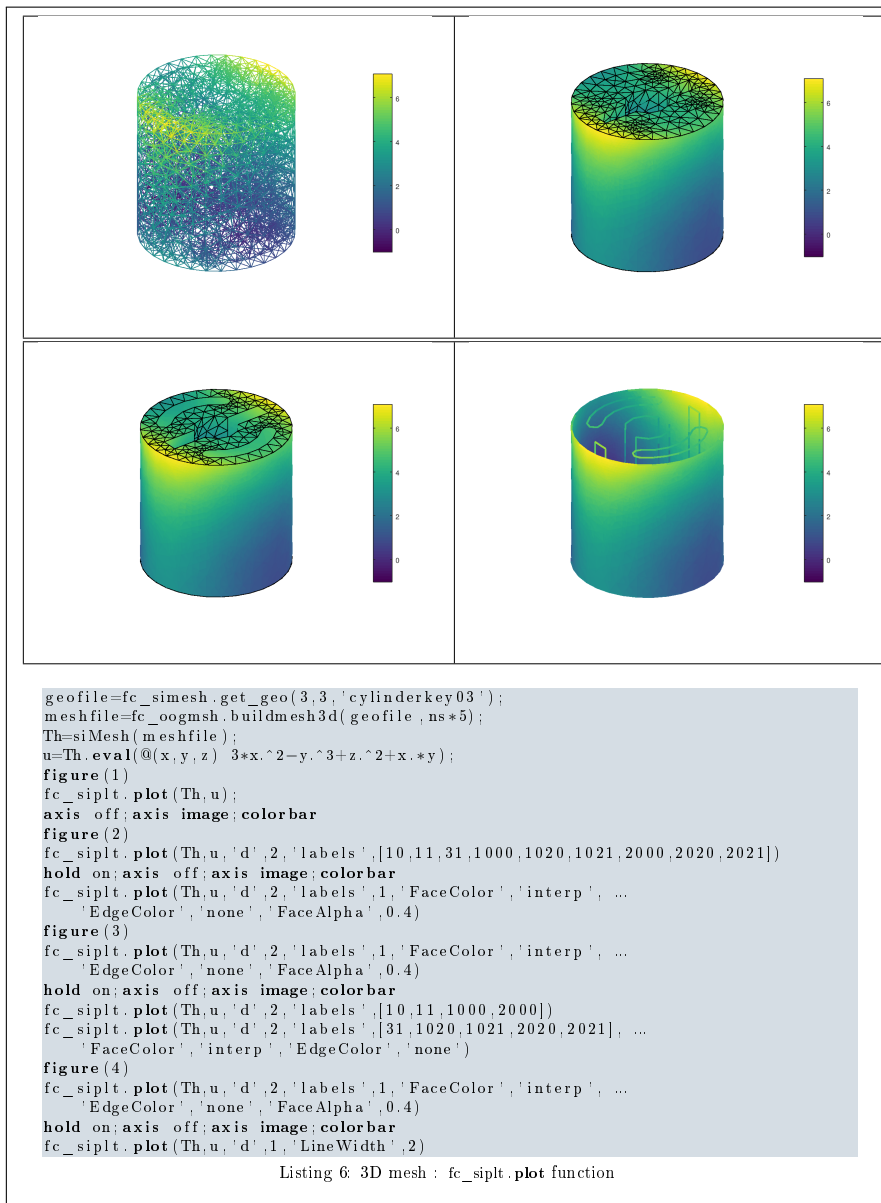
One can use any option of the following functions according to the type of d -simplex.

- In dimension 3, **patch** function is used for $d \in \llbracket 1, 3 \rrbracket$.
- In dimension 2,
 - for $d == 2$, if `'plan'` is true, **patch** function is used, otherwise **trisurf** function,
 - for $d == 1$, **patch** function is used.
- In dimension 1 and $d == 1$, **plot** function is used

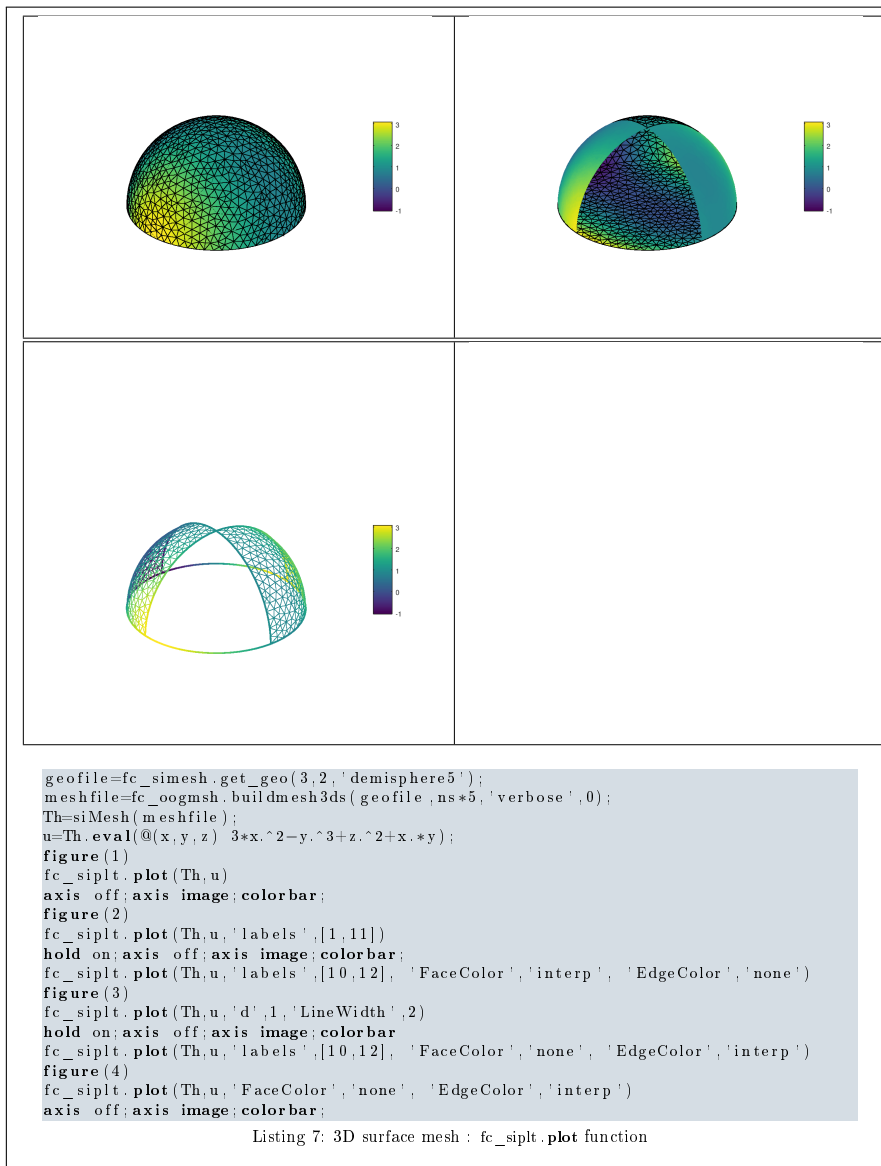
2D example The following example use the `.geo` file `condenser11.geo` which is in the directory `geodir` of the toolbox.



3D example The following example use the *.geo* file *cylinderkey.geo* which is in the directory *geodir* of the toolbox. This file contains description of a 3D mesh with simplices of dimensions 1, 2 and 3.



3D surface example The following example use the *.geo* file *demisphere5.geo* which is in the directory *geodir* of the toolbox. This file contains description of a 3D surface mesh with simplices of dimensions 1 and 2.



6 PLOTISO function

The function `FC_SIPLT.PLOTISO` displays isolines from datas on the mesh or parts of the mesh defined by an `siMESH` object. This function only works with 2-simplices in space dimension 2 or 3.

Syntaxe

```

fc_siplt.plotiso(Th,u)
fc_siplt.plotiso(Th,u,Name,Value,...)

```

Description

`fc_siplt.plotiso(Th,u)` displays data `u` on all the 2-dimensional simplices elements. The data `u` is an 1D-array of size `Th.nq` or `Th.nqGlobal` or `Th.nqParent`.

`fc_siplt.plotiso(Th,u,key,value, ...)` specifies function options using one or more `key,value` pair arguments. Options of first level are

- `'niso'` : to specify the number of isolines (default : 10)
- `'isorange'` : to specify the list of isovalues (default : empty)
- `'isocolorbar'` : if true, colorbar with isovalues is drawn (default : false)
- `'format'` : to specify the format of the isovalues on the colorbar (default : '%g')
- `'labels'` : to select the labels of the elements to display data,
- `'plan'` : if true, (default : false)
- `'color'` : to specify one color for all isolines (default : empty)
- `'mouse'` : if true, display information on clicked isoline (default : false)

The options of second level are all options of

- `plot3` function in dimension 3 or in dimension 2 with `'plan'` set to false
- `plot` function in 2 with `'plan'` set to true

This function accepts until 4 output arguments :

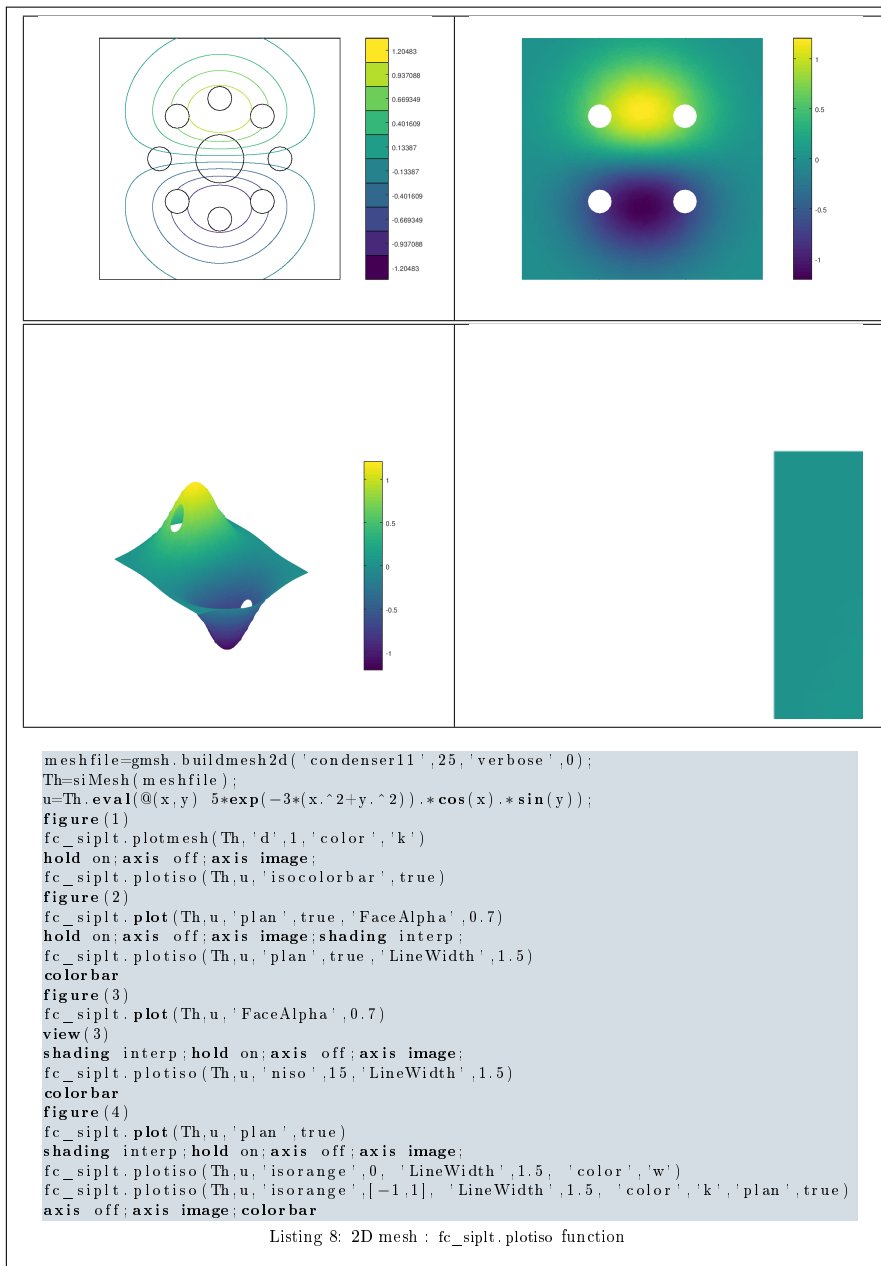
bullet 1st output is the colors of the isolines

bullet 2nd output is the isovalues of the isolines

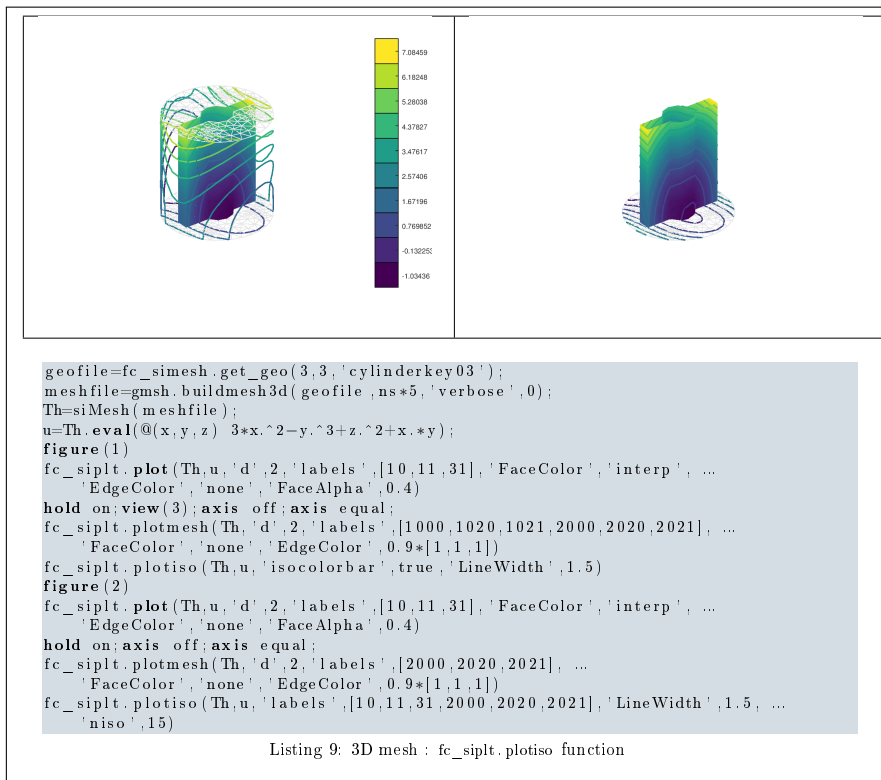
bullet 3th output is the handle of the colobar iso.

bullet 4th output is all the handles of the isolines as an 2D-array of dimension `N`-by-`niso`, where `N` is the number of 2-simplex elementary meshes where isolines are drawn.

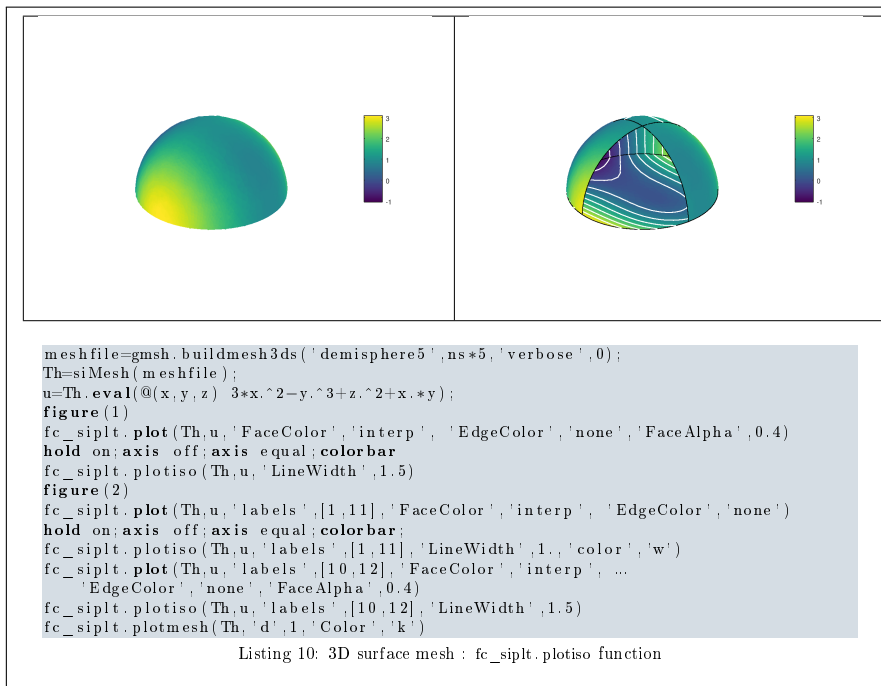
2D example The following example use the `.geo` file `condenser11.geo` which is in the directory `geodir` of the toolbox.



3D example The following example use the *.geo* file *cylinderkey.geo* which is in the directory *geodir* of the toolbox. This file contains description of a 3D mesh with simplices of dimensions 1, 2 and 3.



3D surface example The following example use the *.geo* file *demisphere5.geo* which is in the directory *geodir* of the toolbox. This file contains description of a 3D surface mesh with simplices of dimensions 1 and 2.



7 SLICEMESH function

The function `FC_SIPLT.SLICEMESH` displays intersection of a plane and a 3D mesh or parts of a 3D mesh defined by an `siMESH` object.

Syntaxe

```

fc_siplt.slicemesh(Th,P)
fc_siplt.slicemesh(Th,P,Name,Value,...)

```

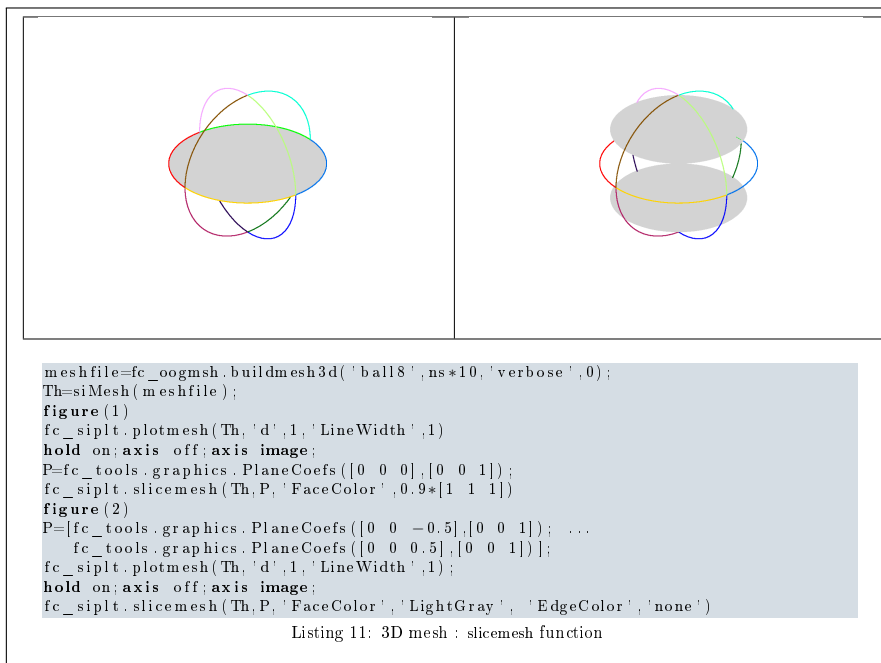
Description

`fc_siplt.slicemesh(Th,P)` displays intersection of the plane defined by $P(1)x + P(2)y + P(3)z + P(4) = 0$ and all the 3-dimensional simplices elements. To compute P one can use the function `PlaneCoefs` of the `FC-SIPLT` package. With this function, the array P , is obtained with $P=PlaneCoefs(Q,V)$ where Q is a point in the plane and V is a vector orthogonal to it.

`fc_siplt.plot(Th,u,Name,Value,...)` specifies function options using one or more `Name,Value` pair arguments. Options of first level are

- `'color'` : to specify the slice color (default : light grey, `rgb=[0.9,0.9,0.9]`)
- `'labels'` : to select the labels of the elements to intersect,

3D example The following example use the *.geo* file `ball18.geo` which is in the directory `geodir` of the toolbox. This file contains description of a 3D mesh with simplices of dimensions 1, 2 and 3.



8 SLICE function

The method `FC_SIPLT.SLICE` displays data on the intersection of a plane and a 3D mesh or parts of a 3D mesh defined by an `SIMESH` object.

Syntaxe

```

fc_siplt.slice(Th,u,P)
fc_siplt.slice(Th,u,P,Name,Value,...)

```

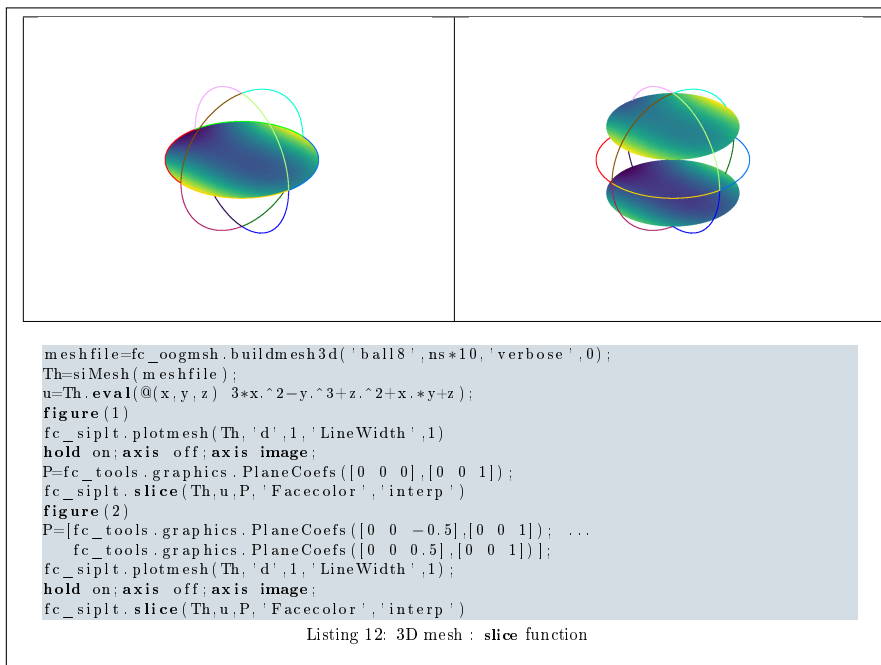
Description

`fc_siplt.slice(Th,u,P)` displays `u` data on the intersection of the plane defined by $P(1)x + P(2)y + P(3)z + P(4) = 0$ and all the 3-dimensional simplices elements. The data `u` is an 1D-array of size `Th.nq` or `Th.nqGlobal` or `Th.nqParent`. To compute `P` one can use the function `PLANECOEFS` of the `FC-TOOLS` package. With this function, the array `P`, is obtained with `P=PlaneCoefs(Q,V)` where `Q` is a point in the plane and `V` is a vector orthogonal to it.

`Th.slice(u,P,Name,Value,...)` specifies function options using one or more `Name,Value` pair arguments. Options of first level are

- `'labels'` : to select the labels of the elements to intersect,

3D example The following example use the *.geo* file `ball18.geo` which is in the directory `geodir` of the toolbox. This file contains description of a 3D mesh with simplices of dimensions 1, 2 and 3.



9

SLICEISO function

The method `FC_SIPLT.SLICEISO` displays isolines of data on the intersection of a plane and a 3D mesh or parts of a 3D mesh defined by an `SI_MESH` object.

Syntaxe

```

fc_siplt.sliceiso(Th,u,P)
fc_siplt.sliceiso(Th,u,P,Name,Value,...)

```

Description

`fc_siplt.sliceiso(Th,u,P)` displays `u` data as isolines on the intersection of the plane defined by $P(1)x + P(2)y + P(3)z + P(4) = 0$ and all the 3-dimensional simplices elements. The data `u` is an 1D-array of size `Th.nq` or `Th.nqGlobal` or `Th.nqParent`. To compute `P` one can use the function `PlaneCoefs` of the `FC-TOOLS` toolbox. With this function, the array `P`, is obtained with `P=fc_tools.graphics.PlaneCoefs(Q,V)` where `Q` is a point in the plane and `V` is a vector orthogonal to the plane.

`fc_siplt.sliceiso(Th,u,P,key,value,...)` allows additional key/value pairs to be used when displaying `u`. The key strings could be

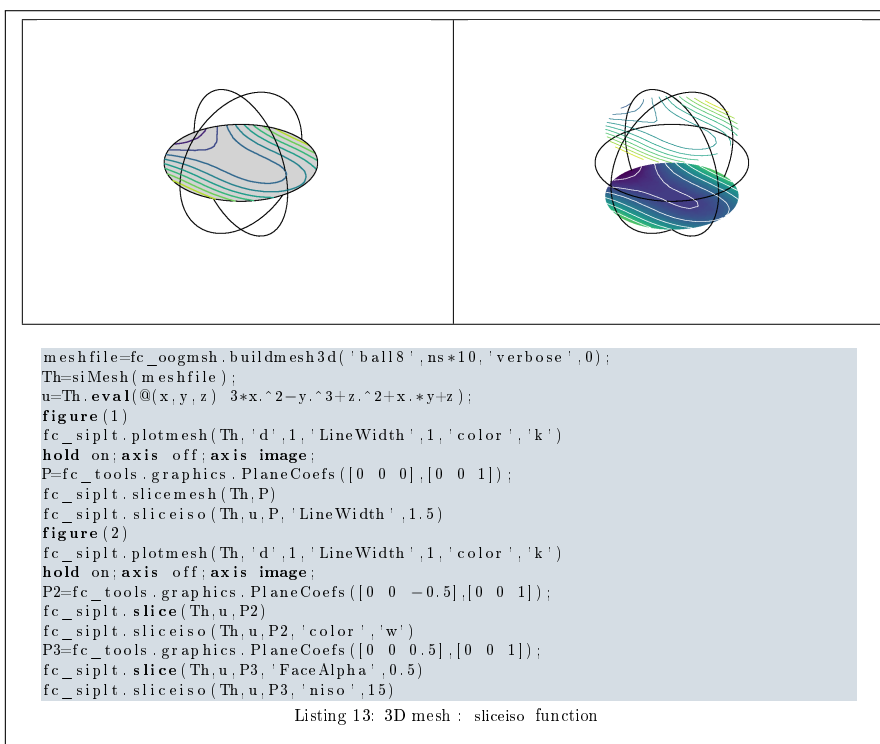
- 'labels' : to select the labels of the elements to intersect,
- 'niso' : to specify the number of isolines (default : 10)
- 'isorange' : to specify the list of isovalues (default : empty)
- 'color' : to specify one color for all isolines (default : empty)
- 'isocolorbar' : if true display a colorbar. Default is false.
- 'format' : to specify the format of the isovalues print in the colorbar. Default is '%g'.

For key strings, one could also used any options of the plot3 function.

This function accepts until 4 output arguments :

- 1st output is the colors of the isolines
- 2nd output is the isovalues of the isolines
- 3th output is the handle of the colobar iso.
- 4th output is all the handles of the isolines as an 2D-array of dimension N-by-niso, where N is the number of elementary meshes where isolines are drawn.

3D example The following example use the *.geo* file `ball18.geo` which is in the directory `geodir` of the toolbox. This file contains description of a 3D mesh with simplices of dimensions 1, 2 and 3.



The method `FC_SIPLT.PLOTQUIVER` displays vector field datas on the mesh or parts of the mesh defined by an `SIMESH` object.

Syntaxe

```
fc_siplt.plotquiver(Th,V)
fc_siplt.plotquiver(Th,V,Key,Value, ...)
```

Description

`fc_siplt.plotquiver(Th,V)` displays vector field U on all the d -dimensional simplices elements in dimension $d = 2$ or $d = 3$. The data V is an 2D-array of size $Th.nq$ -by- d or 2 -by- $Th.nq$.

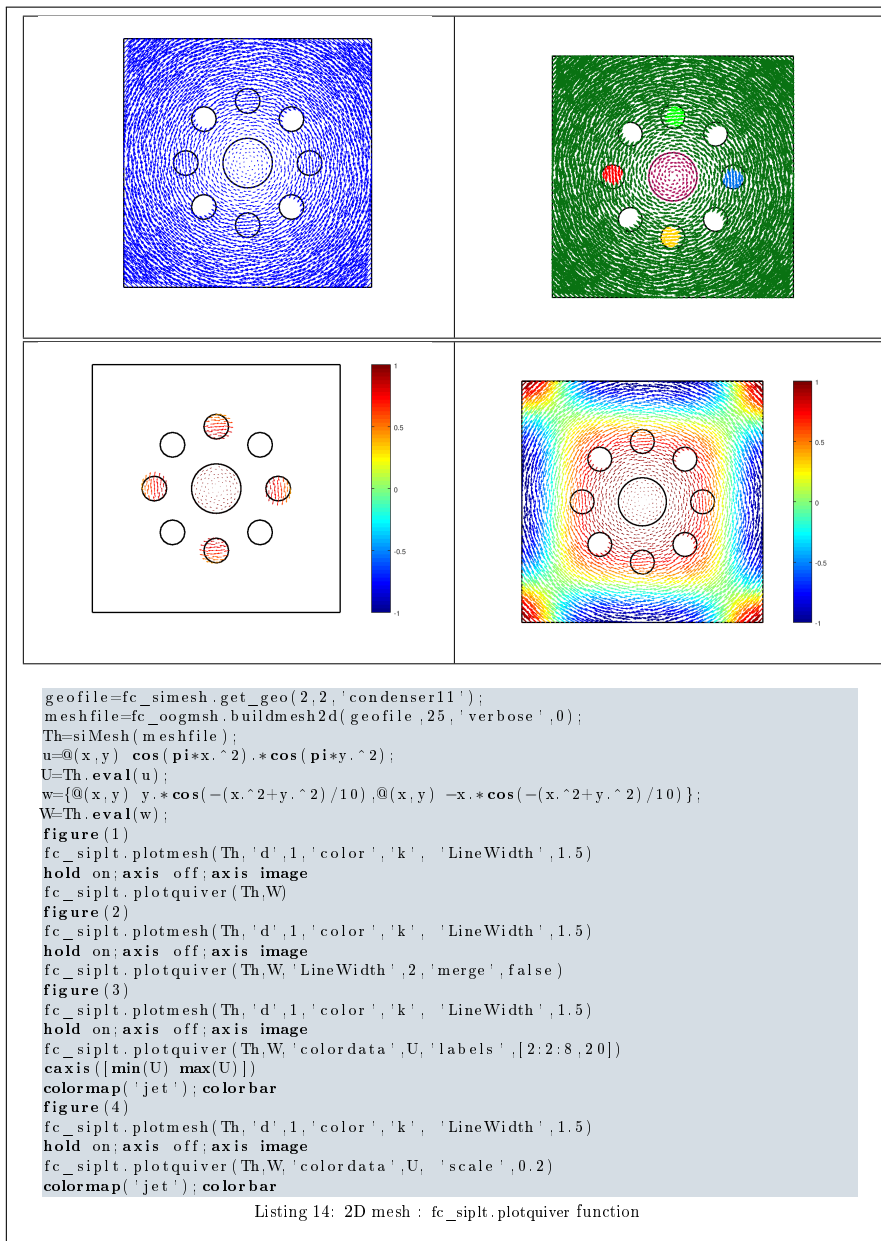
`fc_siplt.plotquiver(Th,V,Key,Value, ...)` specifies function options using one or more `Key,Value` pair arguments. Options of first level are

- `'labels'` : to select the labels of the elements to display data,
- `'freq'` : quiver frequencie, (default : 1)
- `'scale'` : quiver scale, (default : ...)
- `'colordata'` : set colors on each quiver (default : empty).

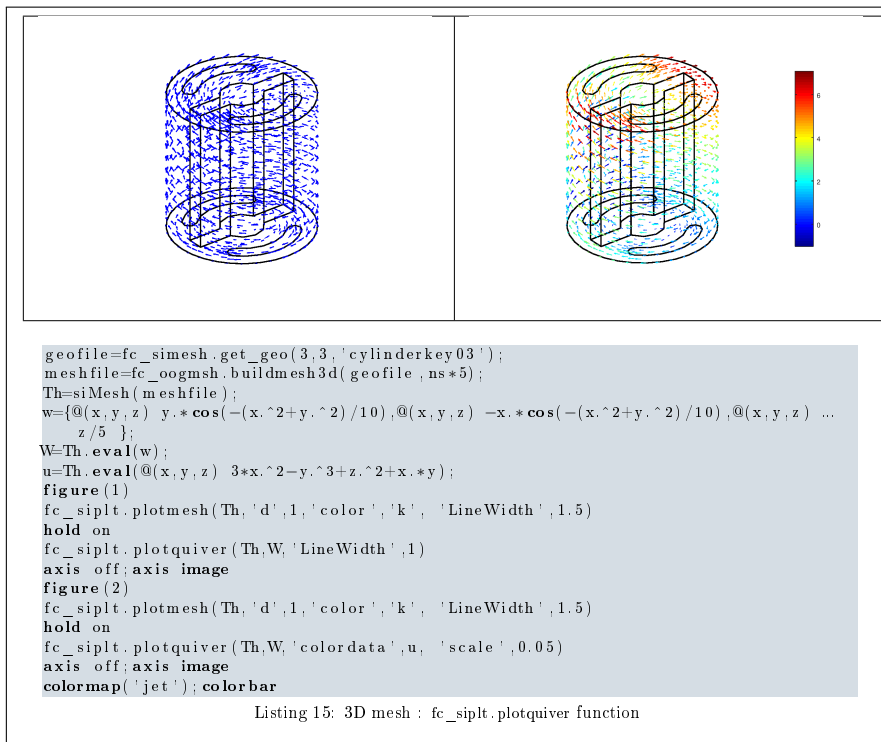
The options of second level depend on space dimension and `'colordata'` option. One can use any option of the following functions

- `quiver` function in dimension 2 with an empty `'colordata'`
- `quiver3` function in dimension 3 with an empty `'colordata'`
- `vfield3` function in dimension 2 or 3 with `'colordata'` set to an 1D-array of length $Th.nq$.

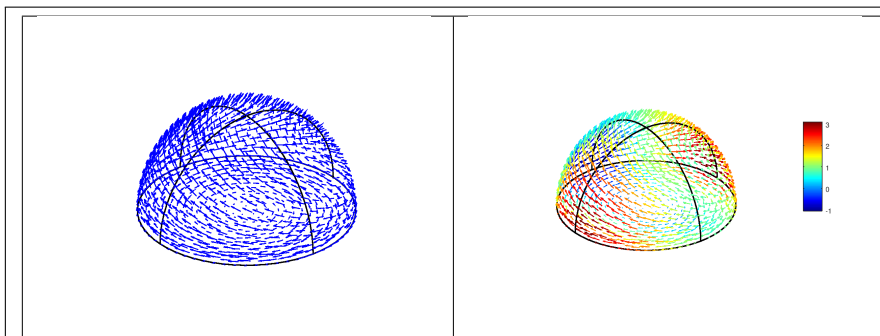
2D example The following example use the `.geo` file `condenser11.geo`.



3D example The following example use the *.geo* file *cylinderkey03.geo*. This file contains description of a 3D mesh with simplices of dimensions 1, 2 and 3.



3D surface example The following example use the *.geo* file *demisphere5.geo* which is in the directory *geodir* of the toolbox. This file contains description of a 3D surface mesh with simplices of dimensions 1 and 2.



```

geofile=fc_simesh.get_geo(3,2,'demisphere5');
meshfile=gmshtool.buildmesh3ds(geofile,ns*5,'verbose',0);
Th=siMesh(meshfile);
w=@(x,y,z) y.*cos(-(x.^2+y.^2)/10),@(x,y,z) -x.*cos(-(x.^2+y.^2)/10),@(x,y,z) z ...
};
W=Th.eval(w);
u=Th.eval(@(x,y,z) 3*x.^2-y.^3+z.^2+x.*y);
figure(1)
fc_siplt.plotmesh(Th,'d',1,'color','k','LineWidth',1.5)
hold on;axis off;axis image
fc_siplt.plotquiver(Th,W,'LineWidth',1)
figure(2)
fc_siplt.plotmesh(Th,'d',1,'color','k','LineWidth',1.5)
hold on;axis off;axis image
fc_siplt.plotquiver(Th,W,'colordata',u,'scale',0.1)
colormap('jet');colorbar

```

Listing 16: 3D surface mesh : `fc_siplt.plotquiver` function