



fc graphics4mesh Matlab toolbox, User's Guide*
version 0.0.4

François Cuvelier[†]

March 22, 2019

Abstract

This Matlab toolbox allows to display simplicial meshes or datas on simplicial meshes. A simplicial mesh must be given by two arrays : the vertices array and the connectivity array.

0 Contents

1	Introduction	2
2	Installation	2
2.1	Installation automatic, all in one (recommanded)	2
2.2	Manual installation	4
3	Mesh	4
4	PLOTMESH function	4
5	PLOT function	8

*L^AT_EX manual, revision 0.0.4, compiled with Matlab 2019a, and toolboxes `fc-graphics4mesh[0.0.4]`, `fc-tools[0.0.27]`, `fc-meshtools[0.1.0]`

[†]LAGA, UMR 7539, CNRS, Université Paris 13 - Sorbonne Paris Cité, Université Paris 8, 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.

6	PLOTISO function	11
7	SLICEMESH function	15
8	SLICE function	16
9	SLICEISO function	17
10	PLOTQUIVER function	20

1 Introduction

The **experimental** Matlab toolbox uses internal functions for displaying simplicial meshes or datas on simplicial meshes. 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 toolbox was tested on various OS with Matlab releases:

Operating system	2015b	2016a	2016b	2017a	2017b	2018a	2018b
CentOS 7.6.1810	✓	✓	✓	✓	✓	✓	✓
Debian 9.8	✓	✓	✓	✓	✓	✓	✓
Fedora 29	✓	✓	✓	✓	✓	✓	✓
OpenSUSE Leap 15.0	✓	✓	✓	✓	✓	✓	✓
Ubuntu 18.04.2 LTS	✓	✓	✓	✓	✓	✓	✓
MacOS High Sierra 10.13.6	✓	✓	✓	✓	✓	✓	✓
MacOS Mojave 10.14							✓
Windows 10 (1803)	✓	✓	✓	✓	✓	✓	✓

It is not compatible with Matlab releases prior to R2015b.

2 Installation

2.1 Installation automatic, all in one (recommanded)

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

`mfc_graphics4mesh_install.m`

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

For example, to install this toolbox in `~/Matlab/toolboxes` directory, one have to copy the file `mfc_graphics4mesh_install.m` in the `~/Matlab/toolboxes` directory. Then in a Matlab terminal run the following commands

```
>> cd ~/Matlab/toolboxes
>> mfc_graphics4mesh_install
```

There is the output of the `mfc_graphics4mesh_install` command on a Linux computer:

```
Parts of the <fc-graphics4mesh> Matlab toolbox.
Copyright (C) 2017-2019 F. Cuvelier

1- Downloading and extracting the toolboxes
2- Setting the <fc-graphics4mesh> toolbox
Write in ...
  ~/Matlab/toolboxes/fc-graphics4mesh-full/fc_graphics4mesh-0.0.4/configure_loc.m ...
  ...
3- Using toolboxes :
  ->          fc-tools : 0.0.26
  ->          fc-mesh  : 0.0.4
*** Using instructions
To use the <fc-graphics4mesh> toolbox:
  addpath('~/Matlab/toolboxes/fc-graphics4mesh-full/fc_graphics4mesh-0.0.4')
  fc_graphics4mesh.init()

See ~/Matlab/toolboxes/mfc_graphics4mesh_set.m
```

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

```
>> addpath('~/Matlab/toolboxes/fc-graphics4mesh-full/fc_meshtools.0.0.4')
>> fc_graphics4mesh.init()
```

If it's the first time the `fc_meshtools.init()` function is used, then its output is

```
Try to use default parameters!
Use fc_tools.configure to configure.
Write in ...
  ~/Matlab/toolboxes/fc-mesh-full/fc_tools-0.0.26/configure_loc.m ...
Using fc_meshtools.0.0.4] with fc_tools[0.0.26].
```

Otherwise, the output of the `fc_meshtools.init()` function is

```
Using fc_meshtools.0.0.4] with fc_tools[0.0.26].
```

For **uninstalling**, one just have to delete directory

```
~/Matlab/toolboxes/fc-graphics4mesh-full
```

2.2 Manual installation

This package uses the `fc_mesh` toolbox. So one has to install it as explain in the dedicated web page.

Thereafter, on the `fc_graphics4mesh` dedicated web page, one can found link to archives (`zip`, `7z` or `tar.gz` format)

- Downloads an archive and extract it on a folder, for example `~/Matlab/toolboxes`. The toolbox path is `~/Matlab/toolboxes/mfc-graphics4mesh-0.0.3`
- Adds the toolbox path in Matlab with `addpath` command.
- Verifies that the `fc_mesh` toolbox is in the Matlab path. Otherwise, adds it...

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-`nq` array where `dim` is the space dimension (2 or 3) and `nq` the number of vertices. The connectivity array `me` is a $(d + 1)$ -by-`nme` array where `nme` 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_meshtools.simplicial.getMesh3D(3)` return a 3-simplicial mesh (*main* mesh) in space dimension $dim = 3$,
- `[q2,me2,toGlobal2]=fc_meshtools.simplicial.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 function `PLOTMESH` displays a mesh given by

Syntaxe

```
fc_graphics4mesh.plotmesh(q,me)
fc_graphics4mesh.plotmesh(q,me,Name,Value,...)
```

Description

`plotmesh(q,me)` displays all the Th.d-dimensional simplices elements.

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

- **'color'** : to specify the color of the displayed mesh elements. (default : 'blue'),
- **'cutPlan'** : (only for simplices in dimension 3) cut mesh by n plans given by n -by-4 array P where the equation of the i -th cut plan is given by

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

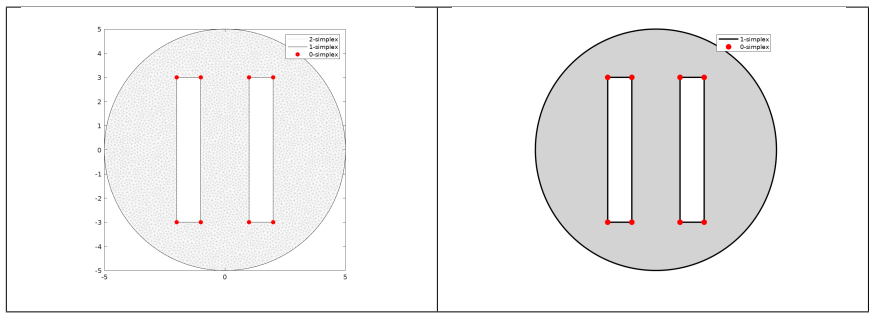
The normal vector $P(i,1 : 3)$ pointed to the part of the mesh not displayed. default : [] (no cut).

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** or **patch** 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

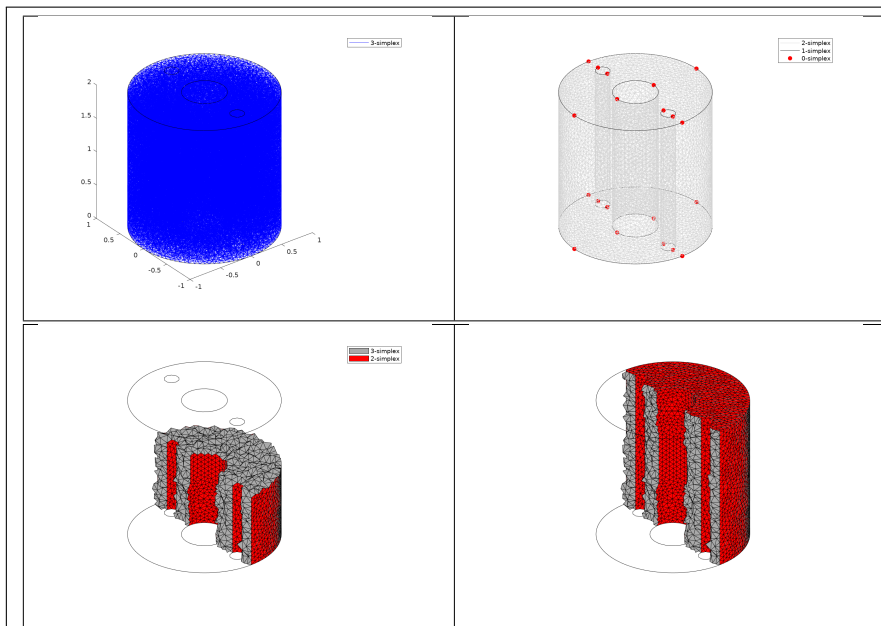


```
[q2,me2,toGlobal2]=fc_meshtools.simplicial.getMesh2D(2);
[q1,me1,toGlobal1]=fc_meshtools.simplicial.getMesh2D(1);
[q0,me0,toGlobal0]=fc_meshtools.simplicial.getMesh2D(0);
figure(1)
fc_graphics4mesh.plotmesh(q2,me2,'Color','LightGray','DisplayName','2-simplex')
hold on
fc_graphics4mesh.plotmesh(q1,me1,'color','black','DisplayName','1-simplex')
fc_graphics4mesh.plotmesh(q0,me0,'color','red','DisplayName','0-simplex')
axis image
legend show

figure(2)
fc_graphics4mesh.plotmesh(q2,me2,'fill',true,'Color','LightGray', ...
'EdgeColor','None')
hold on
fc_graphics4mesh.plotmesh(q1,me1,'color','black','Linewidth',2, ...
'DisplayName','1-simplex')
fc_graphics4mesh.plotmesh(q0,me0,'color','red','markersize',8, ...
'DisplayName','0-simplex')
axis image;axis off
legend show
```

Listing 1: 2D plot mesh

3D example



```

[q3,me3,toGlobal3]=fc_meshtools.simplicial.getMesh3D(3);
[q2,me2,toGlobal2]=fc_meshtools.simplicial.getMesh3D(2);
[q1,me1,toGlobal1]=fc_meshtools.simplicial.getMesh3D(1);
[q0,me0,toGlobal0]=fc_meshtools.simplicial.getMesh3D(0);
figure(1)
view(3)
fc_graphics4mesh.plotmesh(q3,me3,'EdgeColor','blue','FaceColor','None', ...
'DisplayName','3-simplex')
hold on
fc_graphics4mesh.plotmesh(q1,me1,'color','black')
axis image
legend show

figure(2)
view(3)
fc_graphics4mesh.plotmesh(q2,me2,'EdgeColor','LightGray','FaceColor','None', ...
'DisplayName','2-simplex')
hold on
fc_graphics4mesh.plotmesh(q1,me1,'color','black','DisplayName','1-simplex')
fc_graphics4mesh.plotmesh(q0,me0,'color','red','DisplayName','0-simplex')
axis image;axis off
legend show

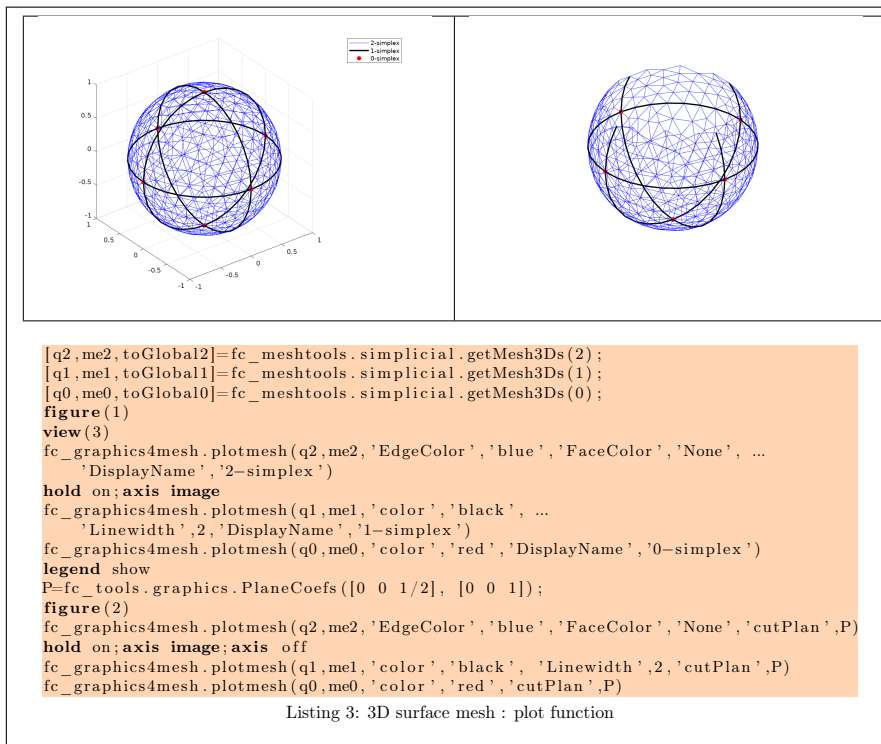
P=[fc_tools.graphics.PlaneCoefs([0 0 1], [0 0 1]); ...
fc_tools.graphics.PlaneCoefs([0 0 1],[-1 0 0])];
figure(3)
fc_graphics4mesh.plotmesh(q1,me1,'color','black')
hold on
fc_graphics4mesh.plotmesh(q3,me3,'cutPlan',P,'Color','DarkGrey', ...
'DisplayName','3-simplex')
fc_graphics4mesh.plotmesh(q2,me2,'cutPlan',P,'Color','red', ...
'DisplayName','2-simplex')
axis image;axis off
legend show

P=fc_tools.graphics.PlaneCoefs([0 0 1], [-1 0 0]);
figure(4)
fc_graphics4mesh.plotmesh(q1,me1,'color','black')
hold on
fc_graphics4mesh.plotmesh(q3,me3,'cutPlan',P,'Color','DarkGrey')
fc_graphics4mesh.plotmesh(q2,me2,'cutPlan',P,'Color','red')
axis image; axis off

```

Listing 2: 3D plot mesh

3D surface example



5 PLOT function

The function `PLOT` displays data on a mesh given by its vertices array `q` and its connectivity array `me`.

Syntax

```

fc_graphics4mesh.plot(q,me,u)
fc_graphics4mesh.plot(q,me,u,Name,Value,...)

```

Description

`plot(q,me,u)` displays data `u` on a simplicial mesh. The data `u` can be an handle function or an array.

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

- `'cutPlan'` : (only for simplices in dimension 3) cut mesh by n plans given by n -by-4 array P where the equation of the i -th cut plan is given by

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

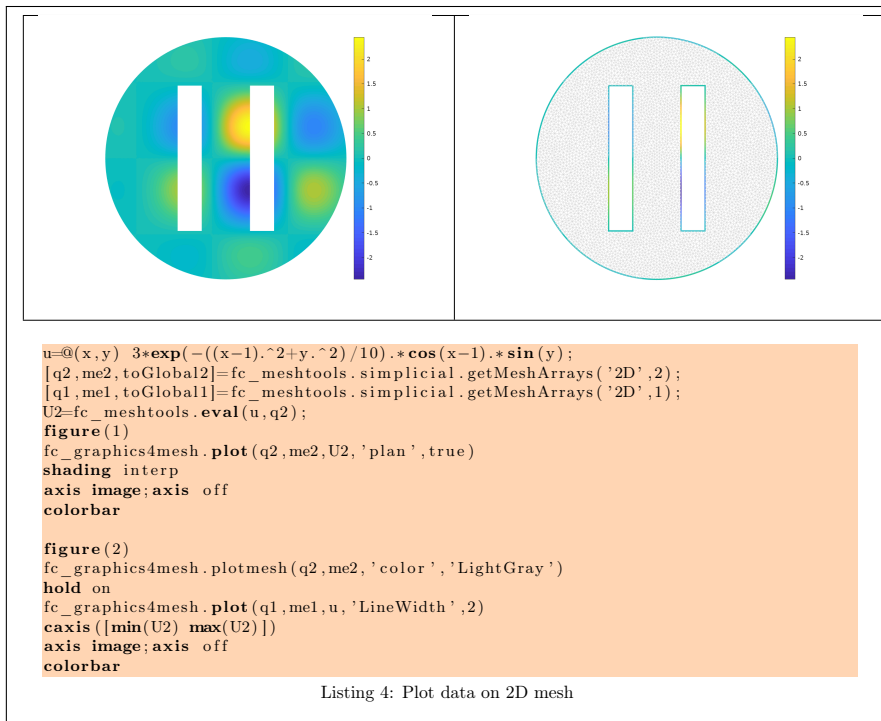
The normal vector $P(i,1 : 3)$ pointed to the part of the mesh not displayed. default : `[]` (no cut).

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

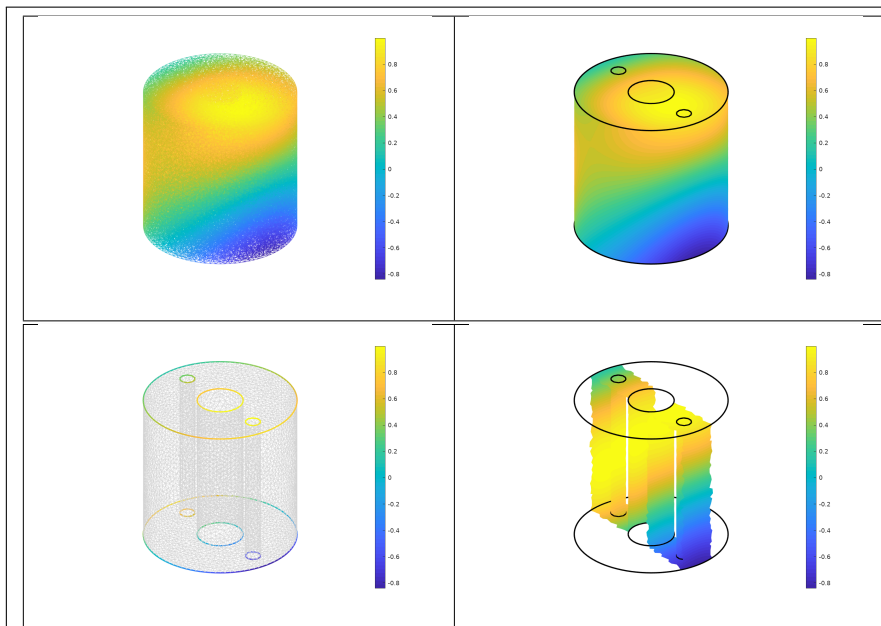
One can use any option of the following functions according to the type of d -simplex to be represented.

- In dimension 3, **patch** function is used.
- In dimension 2,
 - if $d == 2$, **surf** or **patch** (option 'plan' to true) function is used,
 - if $d == 1$, **patch** function is used,
- In dimension 1, **plot** function is used.

2D example



3D example



```

u=@(x,y,z) cos(x).*sin(y+z);
[q3,me3,toGlobal3]=fc_meshtools.simplicial.getMeshArrays('3D',3);
U3=fc_meshtools.eval(u,q3);
[q2,me2,toGlobal2]=fc_meshtools.simplicial.getMeshArrays('3D',2);
[q1,me1,toGlobal1]=fc_meshtools.simplicial.getMeshArrays('3D',1);

figure(1)
fc_graphics4mesh.plot(q3,me3,U3)
shading interp
axis image;axis off
colorbar

figure(2)
fc_graphics4mesh.plot(q2,me2,U3(toGlobal2))
hold on
fc_graphics4mesh.plotmesh(q1,me1,'color','black','LineWidth',2)
shading interp
axis image;axis off
caxis([min(U3),max(U3)])
colorbar

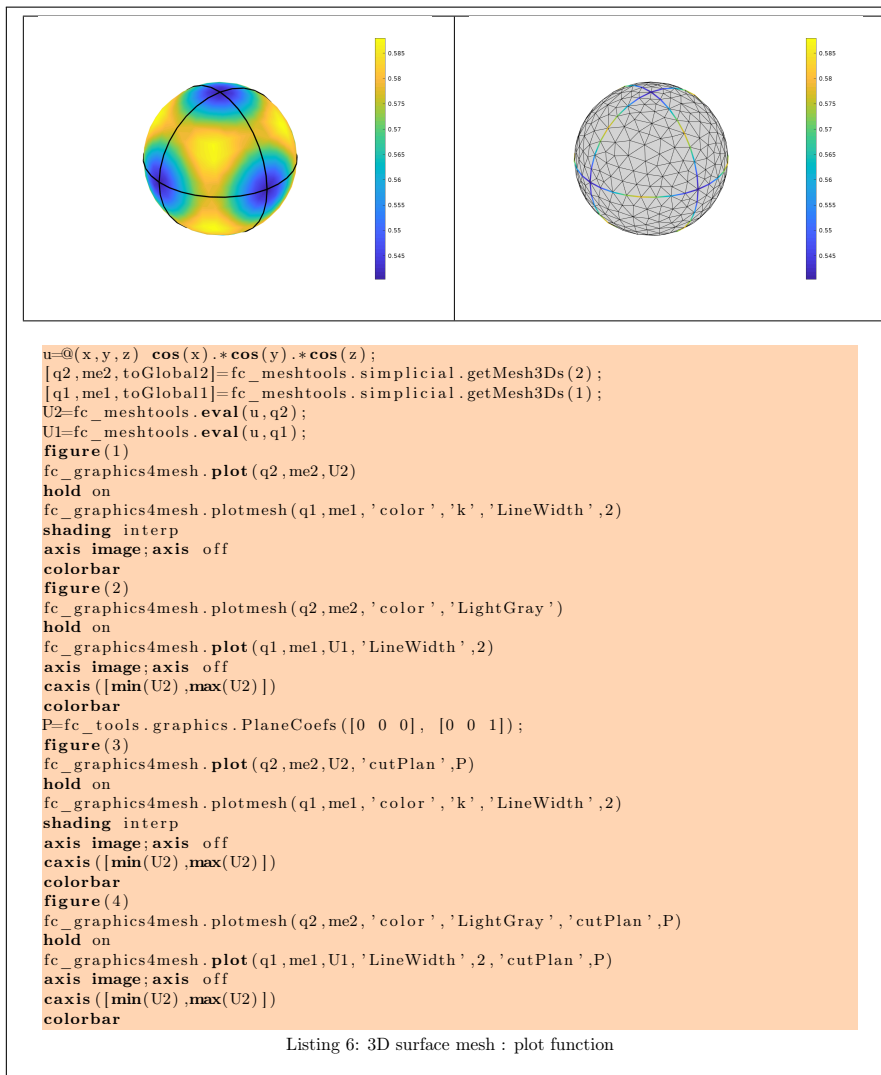
figure(3)
fc_graphics4mesh.plotmesh(q2,me2,'EdgeColor','LightGray','FaceColor','None')
hold on
fc_graphics4mesh.plot(q1,me1,u,'LineWidth',2)
axis image;axis off
caxis([min(U3),max(U3)])
colorbar

P=[fc_tools.graphics.PlaneCoefs([0.2 0 1], [1 0 0]); ...
fc_tools.graphics.PlaneCoefs([-0.2 0 1], [-1 0 0])];
figure(4)
fc_graphics4mesh.plot(q2,me2,U3(toGlobal2),'cutPlan',P)
hold on
fc_graphics4mesh.plotmesh(q1,me1,'color','black','LineWidth',2)
shading interp
axis image;axis off
caxis([min(U3),max(U3)])
colorbar

```

Listing 5: Plot data on 3D mesh

3D surface example



6 PLOTISO function

The function `PLOT` displays isolines from datas on a 2-simplicial mesh given by its vertices array `q` and its connectivity array `me`.

Syntax

```

fc_graphics4mesh.plotiso(q,me,u)
fc_graphics4mesh.plotiso(q,me,u,Name,Value,...)

```

Description

`plotiso(q,me,u)` displays isolines from datas on the 2-simplicial mesh given by the vertices array `q` and the connectivity array `me`. The data `u` can be an

handle function or an array.

`plotiso(q,me,u,Name,Value, ...)` specifies function options using one or more Name,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'`)
- `'plan'` : if true, (default : false)
- `'color'` : to specify one color for all isolines (default : empty)
- `'mouse'` : if true, display information on clicked isoline (default : true)

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

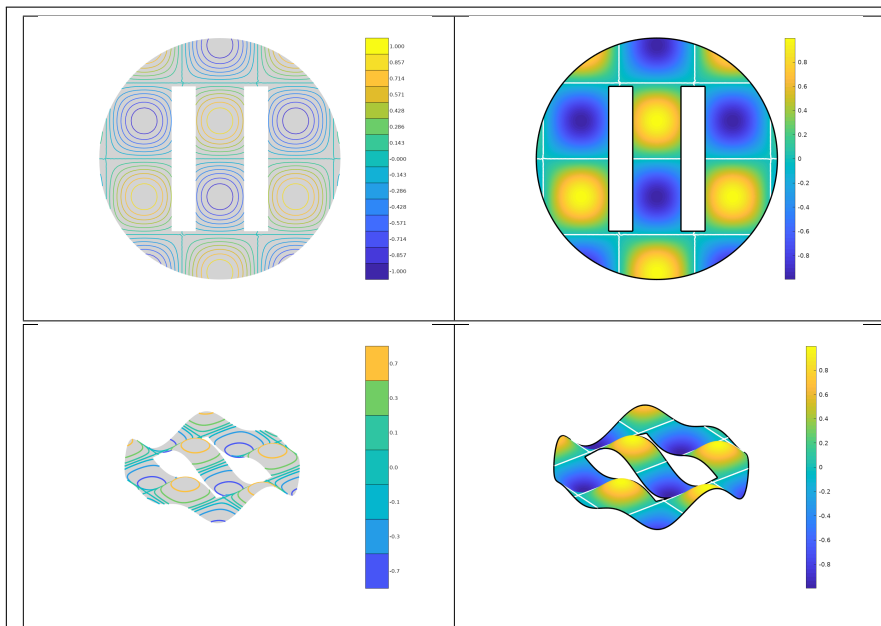
This function accepts until 3 output arguments :

bullet 1st output is the colors of the isolines

bullet 2nd output is the isovalues of the isolines

bullet 3th 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



```

u=@(x,y) cos(x).*sin(y);
[q2,me2,toGlobal2]=fc_meshtools.simplicial.getMesh2D(2);
[q1,me1,toGlobal1]=fc_meshtools.simplicial.getMesh2D(1);
U2=fc_meshtools.eval(u,q2);
U1=fc_meshtools.eval(u,q1);

figure(1)
fc_graphics4mesh.plotmesh(q2,me2,'color','LightGray','fill',true, ...
    'EdgeColor','None','FaceColor','LightGray')
hold on
fc_graphics4mesh.plotiso(q2,me2,U2,'plan',true, ...
    'niso',15,'isocolorbar',true,'format','%3f','LineWidth',1)
axis image;axis off

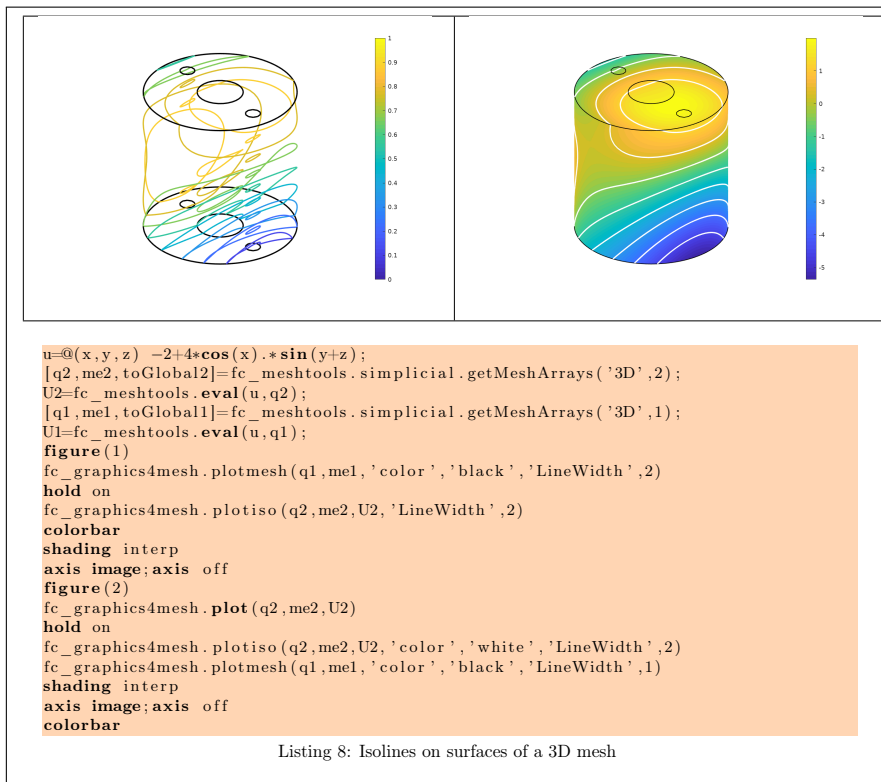
figure(2)
fc_graphics4mesh.plot(q2,me2,U2,'plan',true)
hold on
fc_graphics4mesh.plotiso(q2,me2,U2,'plan',true, ...
    'Color','w','isorange',0,'LineWidth',2)
fc_graphics4mesh.plotmesh(q1,me1,'LineWidth',2,'Color','k')
colorbar
shading interp
axis image;axis off

figure(3)
fc_graphics4mesh.plotmesh(q2,me2,'z',U2,'fill',true,'Color','LightGray')
hold on
isorange=[-0.7,-0.3,-0.1,0,0.1,0.3,0.7];
fc_graphics4mesh.plotiso(q2,me2,U2,'LineWidth',2, ...
    'isorange',isorange,'isocolorbar',true,'format','%1f')
axis image;axis off
figure(4)
fc_graphics4mesh.plot(q2,me2,U2)
hold on
fc_graphics4mesh.plotiso(q2,me2,U2,'LineWidth',2, ...
    'Color','w','isorange',0,'LineWidth',2)
fc_graphics4mesh.plotmesh(q1,me1,'z',U1,'LineWidth',2,'Color','k')
axis image;axis off
shading interp
colorbar

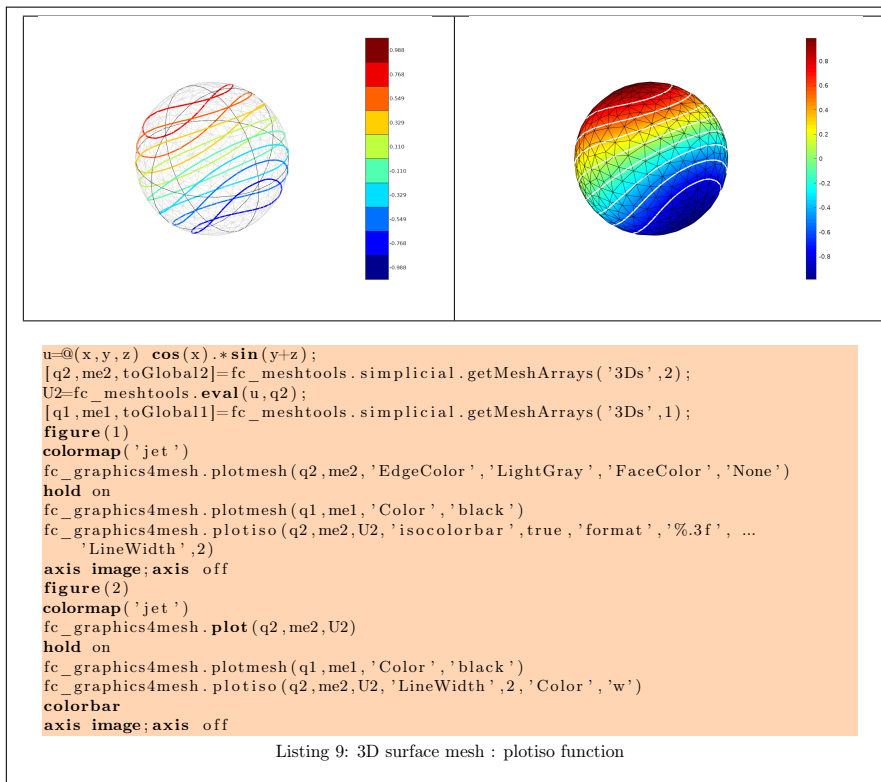
```

Listing 7: Isolines on a 2D mesh

3D example



3D surface example



7 SLICEMESH function

The **SLICEMESH** function displays intersection of a plane and a 3D mesh given by its vertices array q and its connectivity array me .

Syntaxe

```

fc_graphics4mesh.slicemesh(q,me,P)
fc_graphics4mesh.slicemesh(q,me,P,Name,Value,...)

```

Description

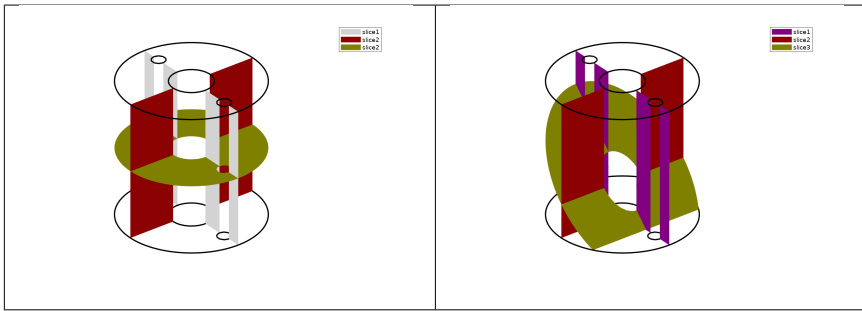
`slicemesh(q,me,P)` displays intersection of the plane defined by $P(1)x + P(2)y + P(3)z + P(4) = 0$ and all the 3-dimensional simplices elements given by q and me arrays. To compute P one can use the `fc_tools.graphics.PlaneCoefs` function of the **FC-TOOLS** toolbox. The 1-by-4 array P , is obtained with $P = \text{fc_tools.graphicsPlaneCoefs}(Q,V)$ where Q is a point in the plane and V is a vector orthogonal to it. One can also use a n -by-4 array P where each line define a plane.

`slicemesh(q,me,P,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 : 'LightGray', rgb=[0.9,0.9,0.9])

The options of second level are all options of the **patch** function except 'FaceColor' and 'EdgeColor'

3D example



```

[q3,me3,toGlobal3]=fc_meshtools.simplicial.getMeshArrays('3D',3);
[q1,me1,toGlobal1]=fc_meshtools.simplicial.getMeshArrays('3D',1);
figure(1)
P=fc_tools.graphics.PlaneCoefs([0 0 1],[1 0 0]);% ...
fc_graphics4mesh.slicemesh(q3,me3,P,'DisplayName','slice1')
hold on
P=fc_tools.graphics.PlaneCoefs([0 0 1],[0 1 0]);% ...
fc_graphics4mesh.slicemesh(q3,me3,P,'Color','DarkRed','DisplayName','slice2')
P=fc_tools.graphics.PlaneCoefs([0 0 1],[0 0 1]);% ...
fc_graphics4mesh.slicemesh(q3,me3,P,'Color','Olive','DisplayName','slice2')
fc_graphics4mesh.plotmesh(q1,me1,'color','k','LineWidth',2)
axis off; axis image
legend('show')
figure(2)
P=[fc_tools.graphics.PlaneCoefs([0 0 1/2],[1 0 0]); ...
fc_tools.graphics.PlaneCoefs([0 0 1/2],[0 1 0]); ...
fc_tools.graphics.PlaneCoefs([0 0 1/2],[0 -1 1])];
fc_graphics4mesh.slicemesh(q3,me3,P,'Color',{'Purple','DarkRed','Olive'}, ...
'DisplayName',{'slice1','slice2','slice3'})
hold on
fc_graphics4mesh.plotmesh(q1,me1,'color','k','LineWidth',2)
axis off; axis image
legend('show')

```

Listing 10: 3D mesh : slicemesh function

8 SLICE function

The **SLICE** function displays intersection of a plane and a 3D mesh given by its vertices array *q* and its connectivity array *me*.

Syntaxe

```

fc_graphics4mesh.slice(q,me,u,P)
fc_graphics4mesh.slice(q,me,u,P,Name,Value,...)

```

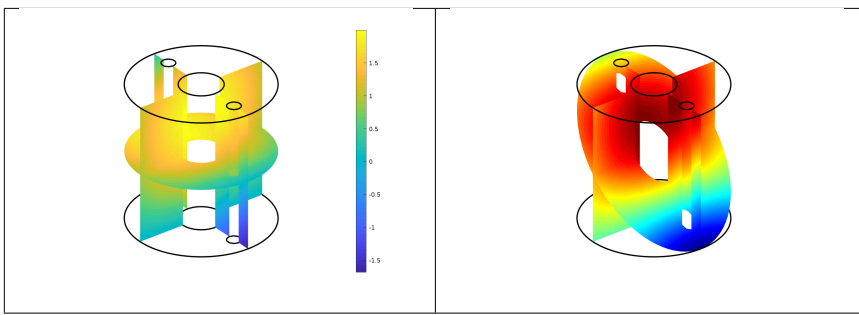
Description

slice(q,me,u,P) displays 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 given by q and me arrays. To compute P one can use the `fc_tools.graphics.PlaneCoefs` function of the **FC-TOOLS** toolbox. 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 it. One can also use a n -by-4 array P where each line define a plane.

`slice(q,me,u,P,Name,Value, ...)` specifies function options using one or more `Name,Value` pair arguments which are those of the **patch** function excepts `'FaceColor'` and `'EdgeColor'`.

3D example



```

u=@(x,y,z) 2*cos(x).*sin(y+z);
[q3,me3,toGlobal3]=fc_meshtools.simplicial.getMeshArrays('3D',3);
U3=fc_meshtools.eval(u,q3);
[q1,me1,toGlobal1]=fc_meshtools.simplicial.getMeshArrays('3D',1);
figure(1)
P=fc_tools.graphics.PlaneCoefs([0 0 1],[1 0 0]);
fc_graphics4mesh.slicemesh(q3,me3,P)
hold on
fc_graphics4mesh.slice(q3,me3,U3,P)
P=fc_tools.graphics.PlaneCoefs([0 0 1],[0 1 0]);
fc_graphics4mesh.slice(q3,me3,U3,P)
P=fc_tools.graphics.PlaneCoefs([0 0 1],[0 0 1]);% ...
fc_graphics4mesh.slice(q3,me3,U3,P)
fc_graphics4mesh.plotmesh(q1,me1,'color','k','LineWidth',2)
axis off; axis image
colorbar
figure(2)
P=[fc_tools.graphics.PlaneCoefs([0 0 1],[1 0 0]); ...
   fc_tools.graphics.PlaneCoefs([0 0 1],[0 1 0]); ...
   fc_tools.graphics.PlaneCoefs([0 0 1],[0 -1 1])];
colormap('jet')
fc_graphics4mesh.slice(q3,me3,u,P)
hold on
fc_graphics4mesh.plotmesh(q1,me1,'color','k','LineWidth',2)
axis off; axis image

```

Listing 11: 3D mesh : slice function

9 SLICEISO function

The **SLICEISO** function displays isolines of datas on the intersection of a plane and a 3D mesh given by its vertices array q and its connectivity array me .

Syntaxe

```
fc_graphics4mesh.sliceiso(q,me,u,P)
fc_graphics4mesh.sliceiso(q,me,u,P,Name,Value, ...)
```

Description

`sliceiso(q,me,u,P)` displays isolines of data u 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 given by q and me arrays. To compute P one can use the `fc_tools.graphics.PlaneCoefs` function of the **FC-TOOLS** toolbox. The 1-by-4 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 it. One can also used a n -by-4 array P where each line define a plane.

`sliceiso(q,me,u,P,Name,Value, ...)` allows additional key/value pairs to be used when displaying u . The key strings could be

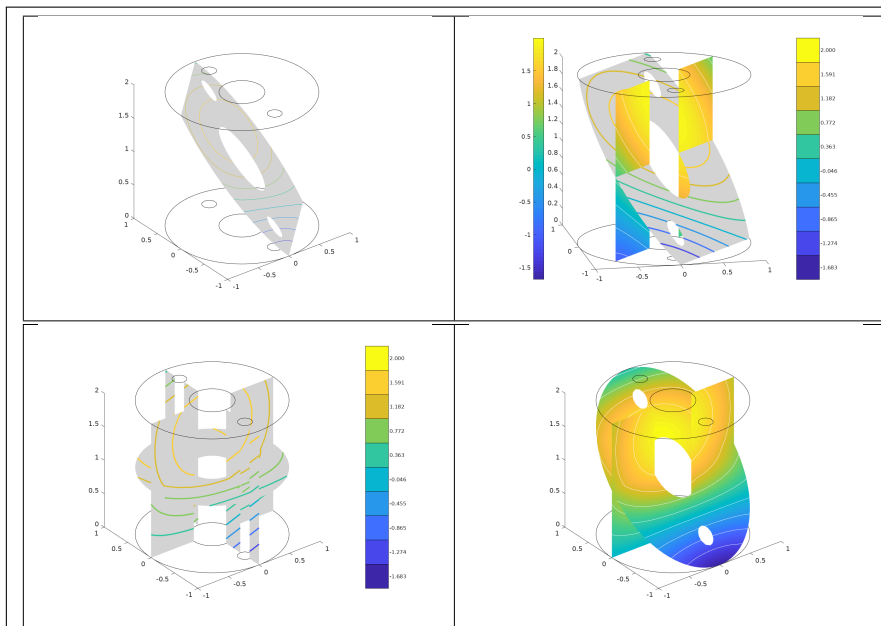
- `'niso'` : to specify the number of isolines (default : 10)
- `'isorange'` : to specify the list of isovalues (default : empty)
- `'color'` : to specify one color for all isolines (default : empty)
- `'isocolorbar'` : if true display a colorbar.Default is false.
- `'format'` : to specify the format of the isovalues print in the colorbar. Default is `'%g'`.
- `'mouse'` : if true, display information on clicked isoline (default : true)

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



```

u=@(x,y,z) 2*cos(x).*sin(y+z);
[q3,me3,toGlobal3]=fc_meshtools.simplicial.getMeshArrays('3D',3);
U3=fc_meshtools.eval(u,q3);
[q1,me1,toGlobal1]=fc_meshtools.simplicial.getMeshArrays('3D',1);
figure (1)
P=fc_tools.graphics.PlaneCoefs([0 0 1],[1 -1 1]);
fc_graphics4mesh.slicemesh(q3,me3,P)
hold on;axis equal;axis image
fc_graphics4mesh.plotmesh(q1,me1,'color','k')
fc_graphics4mesh.sliceiso(q3,me3,U3,P)
figure (2)
fc_graphics4mesh.slicemesh(q3,me3,P)
hold on;axis equal;axis image
fc_graphics4mesh.plotmesh(q1,me1,'color','k')
fc_graphics4mesh.sliceiso(q3,me3,U3,P,'Linewidth',2, ...
    'isocolorbar',true,'format','%3f');
P=fc_tools.graphics.PlaneCoefs([0 0 1],[1 -1 0]);
fc_graphics4mesh.slice(q3,me3,U3,P)
fc_graphics4mesh.sliceiso(q3,me3,U3,P,'color','w');
colorbar('Location','westoutside')
caxis([min(U3),max(U3)]);view(-11,15)
figure (3)
P=[fc_tools.graphics.PlaneCoefs([0 0 1],[1 0 0]); ...
    fc_tools.graphics.PlaneCoefs([0 0 1],[0 1 0]); ...
    fc_tools.graphics.PlaneCoefs([0 0 1],[0 0 1])];
fc_graphics4mesh.slicemesh(q3,me3,P)
hold on;axis equal;axis image
fc_graphics4mesh.plotmesh(q1,me1,'color','k')
fc_graphics4mesh.sliceiso(q3,me3,U3,P,'LineWidth',2, ...
    'isocolorbar',true,'format','%3f');
figure (4)
P=[fc_tools.graphics.PlaneCoefs([0 0 1],[0 1 0]); ...
    fc_tools.graphics.PlaneCoefs([0 0 1],[0 -1 1])];
fc_graphics4mesh.slice(q3,me3,U3,P)
hold on;axis equal;axis image
fc_graphics4mesh.plotmesh(q1,me1,'color','k')
fc_graphics4mesh.sliceiso(q3,me3,U3,P,'Color','w');

```

Listing 12: 3D mesh : sliceiso function

10 `PLOTQUIVER` function

The function `PLOTQUIVER` displays vector field datas on a mesh given by its vertices array `q` and its connectivity array `me`.

Syntax

```
fc_graphics4mesh.plotquiver(q,me,V)
fc_graphics4mesh.plotquiver(q,me,V,Name,Value, ...)
```

Description

`plotquiver(q,me,V)` displays vector field `u` on a simplicial mesh. The vector field data `u` can be a 1-by-dim cell arrays of handle functions or an dim-by- n_q array.

`plotquiver(q,me,V,Name,Value, ...)` specifies function options using one or more Name,Value pair arguments. Options of first level are

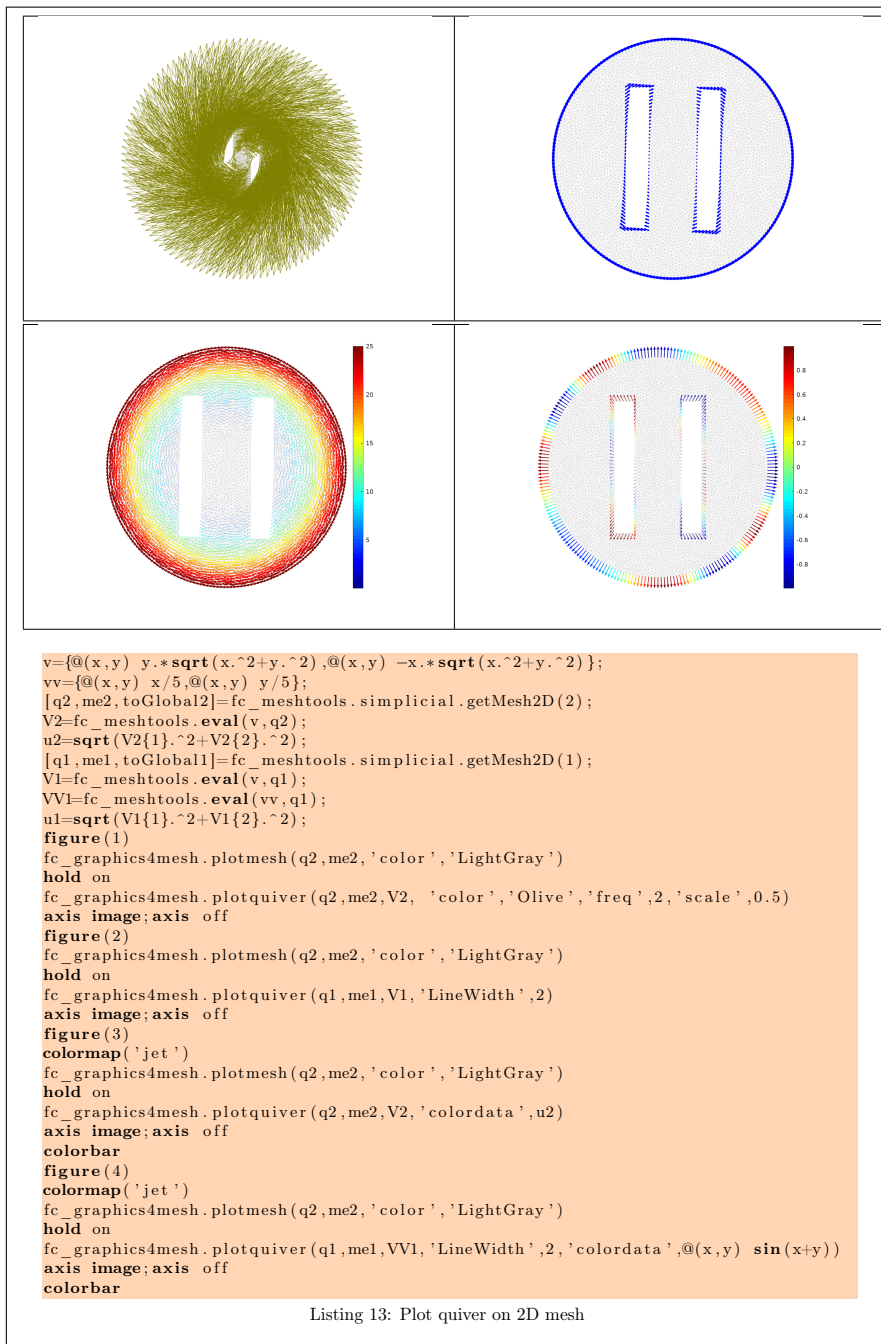
- `'freq'` : quiver frequencie, (default : 1)
- `'scale'` : quiver scale, (default is `fc_meshtools.getCharacteristicLength(q)/20`)
- `'color'` : set one color for all quivers (default: default color of the `quiver` or `quiver3` functions). Cannot be used with `'colordata'` option.
- `'colordata'` : each quiver is colored with a 1-by- n_q array or a handle function (it will evaluated in all vertices) (default : empty).

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

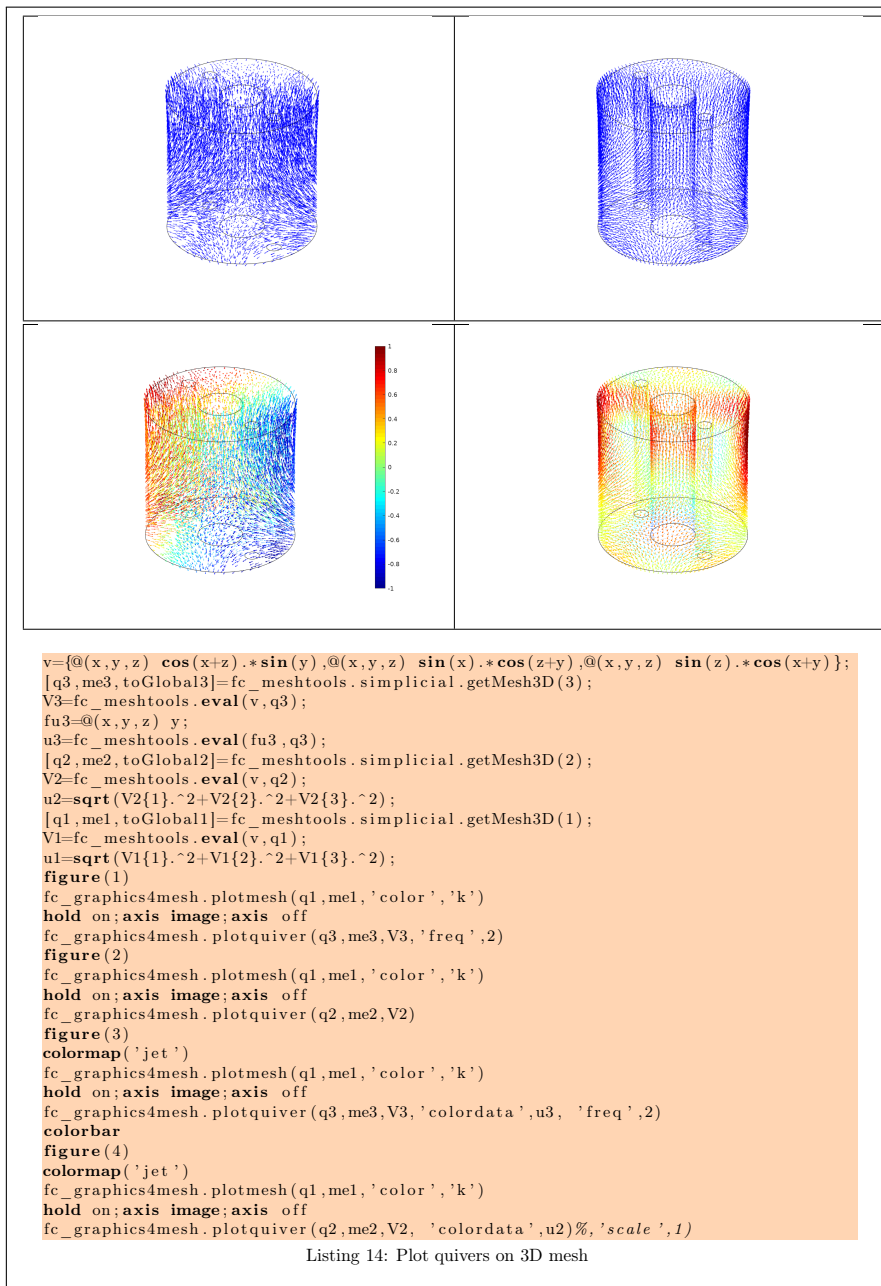
One can use any option of the following functions according to the type of d -simplex to be represented.

- In dimension 3 and with empty `'colordata'` , the `quiver3` function is used.
- In dimension 2 and with empty `'colordata'` , the `quiver` function is used.
- In dimension 2 or 3 and with no empty `'colordata'`, the third party `fc_tools.graphics.vfield3.vfield3` function is used.

2D example



3D example



3D surface example

