



**fc graphics4mesh** Octave package, User's Guide\*  
version 0.1.2

François Cuvelier<sup>†</sup>

March 7, 2020

**Abstract**

This Octave package 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**

---

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Installation</b>	<b>3</b>
2.1	Installation automatic, all in one . . . . .	3
<b>3</b>	<b>Mesh</b>	<b>4</b>
<b>4</b>	<b><code>fc_graphics4mesh.plotmesh</code> function</b>	<b>5</b>
<b>5</b>	<b><code>fc_graphics4mesh.plot</code> function</b>	<b>9</b>

\* $\text{\LaTeX}$  manual, revision 0.1.2, compiled with Octave 5.2.0, and packages `fc-graphics4mesh`[0.1.2], `fc-tools`[0.0.31], `fc-bench`[0.1.2], `fc-amat`[0.1.2], `fc-meshtools`[0.1.3]

<sup>†</sup>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	<code>fc_graphics4mesh.plotiso</code> function	13
7	<code>fc_graphics4mesh.slicemesh</code> function	16
8	<code>fc_graphics4mesh.slice</code> function	17
9	<code>fc_graphics4mesh.sliceiso</code> function	18
10	<code>fc_graphics4mesh.plotquiver</code> function	21
11	<code>fc_graphics4mesh.plotnodes</code> function	24
12	<code>fc_graphics4mesh.plotnodesidx</code> function	27
13	<code>fc_graphics4mesh.plotelementsidx</code> function	30

## 1 Introduction

The **experimental** Octave package 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 package was tested on various OS with Octave releases:

Operating system	4.4.0	4.4.1	5.1.0	5.2.0
CentOS 7.7.1908	✓	✓	✓	✓
Debian 9.11	✓	✓	✓	✓
Fedora 29	✓	✓	✓	✓
OpenSUSE Leap 15.0	✓	✓	✓	✓
Ubuntu 18.04.3 LTS	✓	✓	✓	✓
MacOS High Sierra 10.13.6		✓	✓	✓
MacOS Mojave 10.14.4		✓	✓	✓
MacOS Catalina 10.15.2		✓	✓	✓
Windows 10 (1909)	✓	✓	✓	✓

It is not compatible with Octave releases prior to 4.2.0. Here are the links used to install the Octave releases tested:

- **Linux** : sources from <https://www.gnu.org/software/octave/>;

- **MacOS** : release 4.4.1 installed with dmg file from <http://octave-app.org/Download.html>, releases 5.1.0 and 5.2.0 installed with Homebrew;
- **Windows** : binaries from <https://www.gnu.org/software/octave/>.

## 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 packages (*fc-tools* and *fc-mesh*) in the current directory.

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

```
mkdir -p ~/Octave
cd ~/Octave
HTTP=http://www.math.univ-paris13.fr/~cuvelier/software/codes/Octave
wget $HTTP/fc-graphics4mesh/0.1.1/ofc_graphics4mesh_install.m
```

Then in a Octave terminal run the following commands

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

There is the output of the `ofc_graphics4mesh_install` command:

```
Parts of the <fc-graphics4mesh> Octave package.
Copyright (C) 2017-2020 F. Cuvelier

1- Downloading and extracting the packages
2- Setting the <fc-graphics4mesh> package
Write in ...
  ~/Octave/fc-graphics4mesh-full/fc_graphics4mesh-0.1.2/configure_loc.m ...
  ...
3- Using packages :
->          fc-tools : 0.0.31
->          fc-bench : 0.1.2
->          fc-amat  : 0.1.2
->          fc-meshtools : 0.1.3
with      fc-graphics4mesh : 0.1.2
*** Using instructions
To use the <fc-graphics4mesh> package:
addpath('~/Octave/fc-graphics4mesh-full/fc_graphics4mesh-0.1.2')
fc_graphics4mesh.init()

See ~/Octave/ofc_graphics4mesh_set.m
```

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

```
>> addpath('~\Octave\fc-graphics4mesh-full\ofc-graphics4mesh-0.1.1')
>> fc_graphics4mesh.init()
```

If it's the first time the `fc_graphics4mesh.init()` function is used, then its output is

```
Try to use default parameters!
Use fc_tools.configure to configure.
Write in ...
  ~\Octave\fc-graphics4mesh-full\fc_tools-0.0.31\configure_loc.m ...
Try to use default parameters!
Use fc_bench.configure to configure.
Write in ~\Octave\fc-graphics4mesh-full\fc_bench-0.1.2\configure_loc.m ...
...
Try to use default parameters!
Use fc_amat.configure to configure.
Write in ~\Octave\fc-graphics4mesh-full\fc_amat-0.1.2\configure_loc.m ...
Try to use default parameters!
Use fc_meshtools.configure to configure.
Write in ...
  ~\Octave\fc-graphics4mesh-full\fc_meshtools-0.1.3\configure_loc.m ...
Using fc_graphics4mesh[0.1.2] with fc_tools[0.0.31], fc_bench[0.1.2], ...
      fc_amat[0.1.2], fc_meshtools[0.1.3].
```

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

```
Using fc_graphics4mesh[0.1.1] with fc_tools[0.0.30], fc_bench[0.1.2], ...
      fc_amat[0.1.2], fc_meshtools[0.1.3].
```

For **uninstalling**, one just have to delete directory `~\Octave\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 these 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 fc\_graphics4mesh.plotmesh function

The function fc\_graphics4mesh.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'),
- 'cutPlane' : (only for simplices in dimension 3) cut mesh by  $n$  planes given by  $n$ -by-4 array  $P$  where the equation of the  $i$ -th cut plane 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 :  $\parallel$  (no cut).

- 'inLegend' : to add this mesh in a legend if true. Default is false.
- 'DisplayName' : to specify the name used in a legend for this mesh. Then the ('inLegend' option is forced to true.

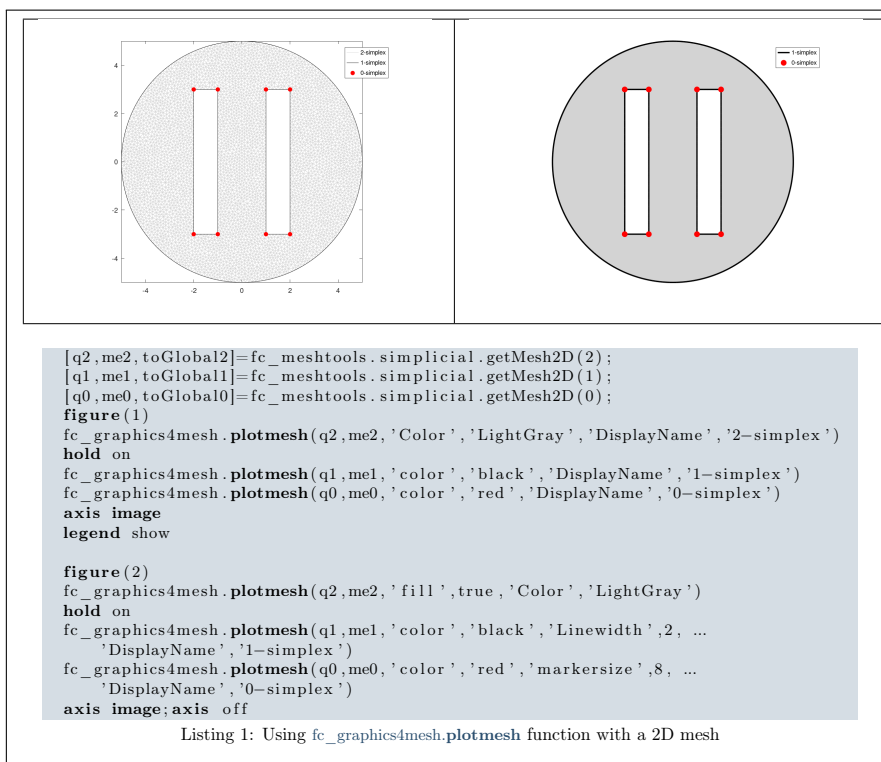
The options of second level depend on the type of elementaries mesh elements to represent.

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

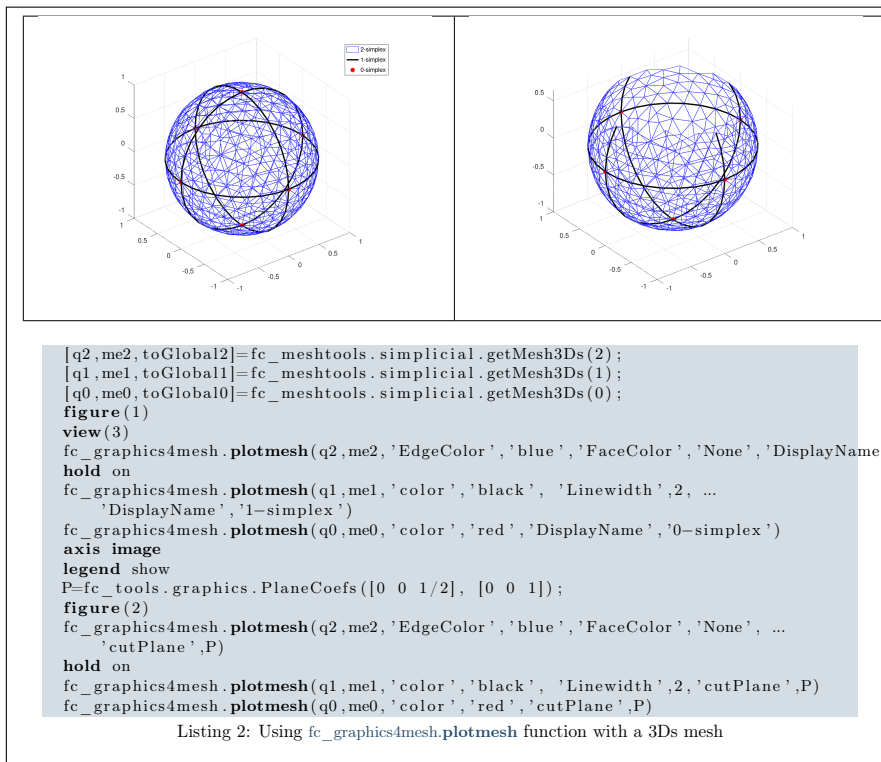
- In dimension 3,
  - if  $d == 3$ , patch function is used,
  - if  $d == 2$ , trimesh function is used,
  - if  $d == 1$ , plot3 function is used,
  - if  $d == 0$ , plot3 function is used,

- 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,

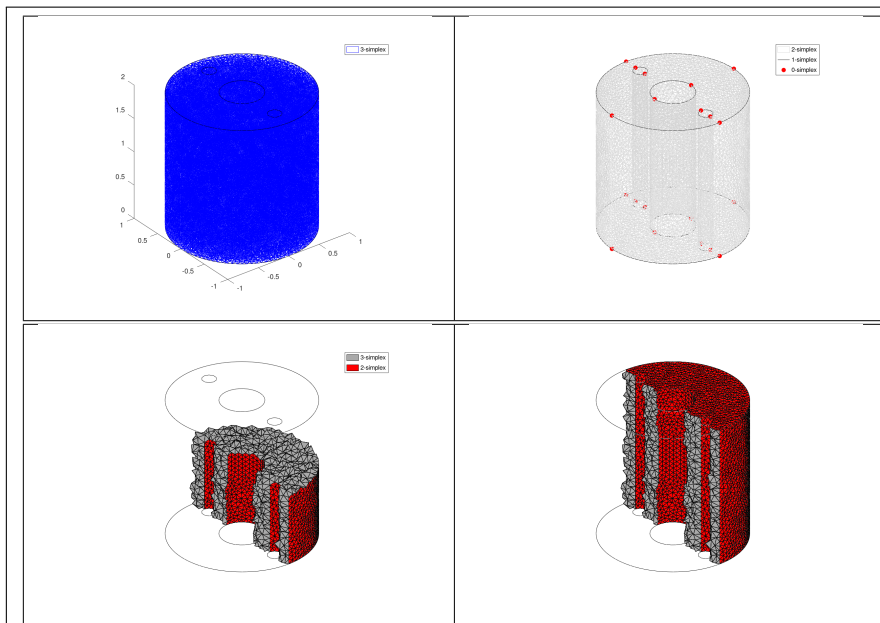
**2D example :** the following code is part of the `fc_graphics4mesh.demos.plotmesh2D` function.



**3Ds example :** the following code is part of the `fc_graphics4mesh.demos.plotmesh3Ds` function.



**3D example :** the following code is part of the `fc_graphics4mesh.demos.plotmesh3D` function.



```

[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,'cutPlane',P,'Color','DarkGrey',...
'DisplayName','3-simplex')
fc_graphics4mesh.plotmesh(q2,me2,'cutPlane',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,'cutPlane',P,'Color','DarkGrey')
fc_graphics4mesh.plotmesh(q2,me2,'cutPlane',P,'Color','red')

```

Listing 3: Using `fc_graphics4mesh.plotmesh` function with a 3D mesh



## 5 `fc_graphics4mesh.plot` function

The function `fc_graphics4mesh.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

- `'cutPlane'` : (only for simplices in dimension 3) cut mesh by  $n$  planes given by  $n$ -by-4 array  $P$  where the equation of the  $i$ -th cut plane is given by

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

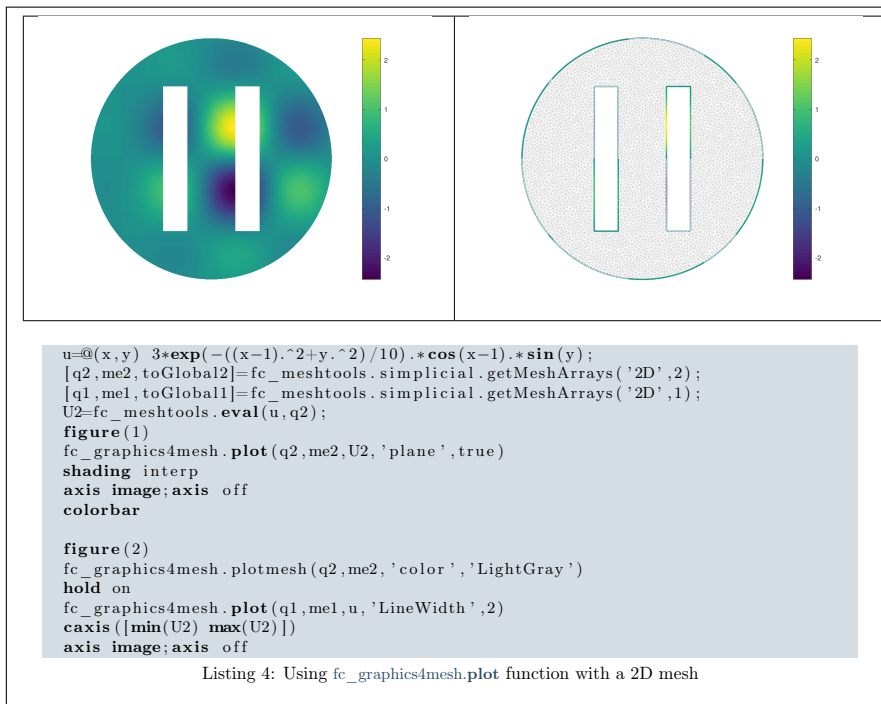
The normal vector  $P(i,1 : 3)$  pointed to the part of the mesh not displayed. default : `[]` (no cut).

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

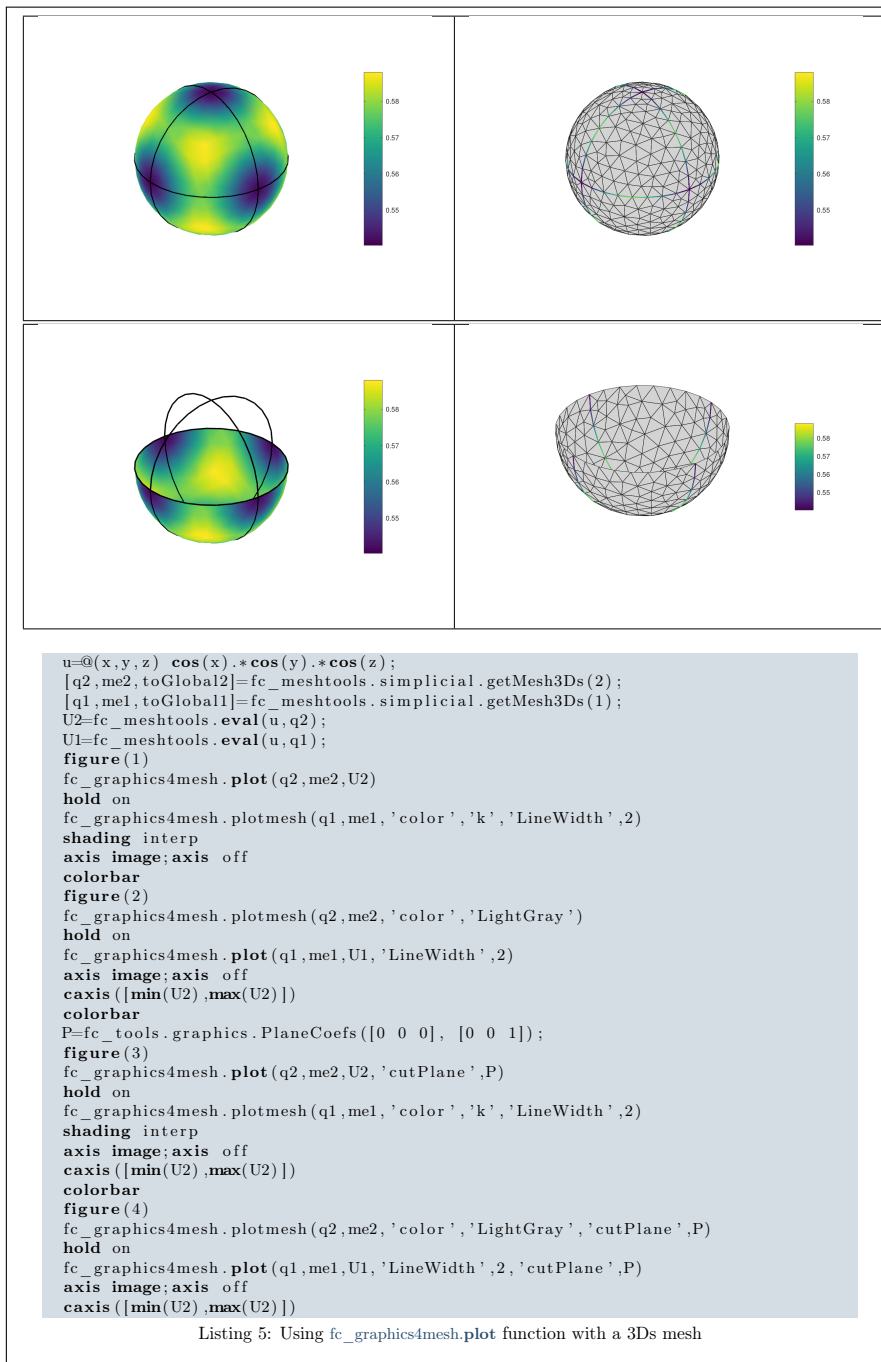
One can use any option of the following functions according to the type of  $d$ -simplex to be represented.

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

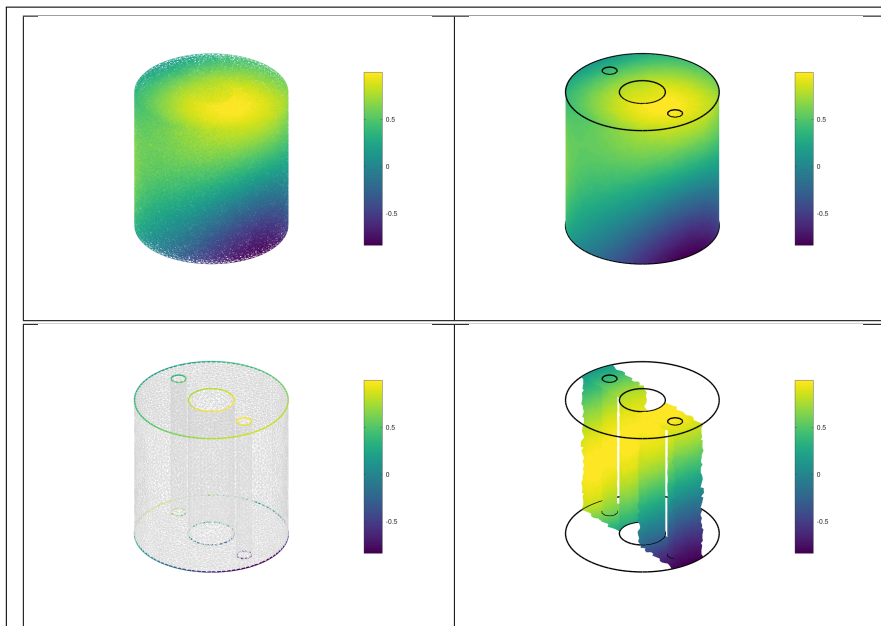
**2D example :** the following code is part of the `fc_graphics4mesh.demos.plot2D` function.



**3Ds example :** the following code is part of the fc\_graphics4mesh.demos.plot3Ds function.



**3D example :** the following code is part of the `fc_graphics4mesh.demos.plot3D` function.



```

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),'cutPlane',P)
hold on
fc_graphics4mesh.plotmesh(q1,me1,'color','black','LineWidth',2)
shading interp
axis image;axis off
caxis([min(U3),max(U3)])

```

Listing 6: Using `fc_graphics4mesh.plot` function with a 3D mesh

## 6 `fc_graphics4mesh.plotiso` function

The function `fc_graphics4mesh.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'`)
- `'plane'` : if `true`, isolines are in the  $xy$ -plane, otherwise isolines are in 3D with  $z$ -value set to `u` (default : `false`)
- `'color'` : to specify one color for all isolines (default : empty)

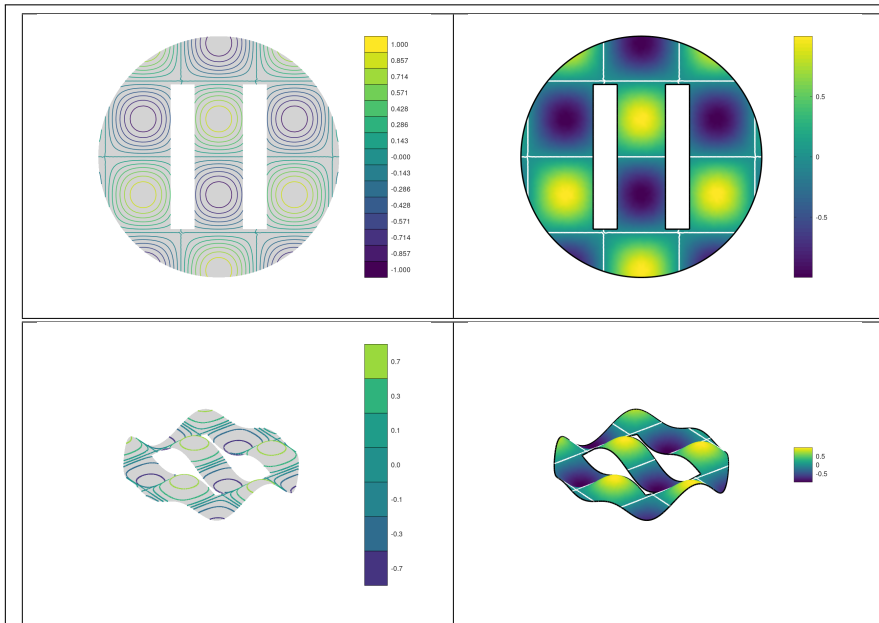
The options of second level are all options of

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

This function accepts until 3 output arguments :

- 1st output is the colors of the isolines
- 2nd output is the isovalues of the isolines
- 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 :** the following code is part of the `fc_graphics4mesh.demos.plotiso2D` function.



```

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,'plane',true,...
'niso',15,'isocolorbar',true,'format','%3f','LineWidth',1)
axis image;axis off

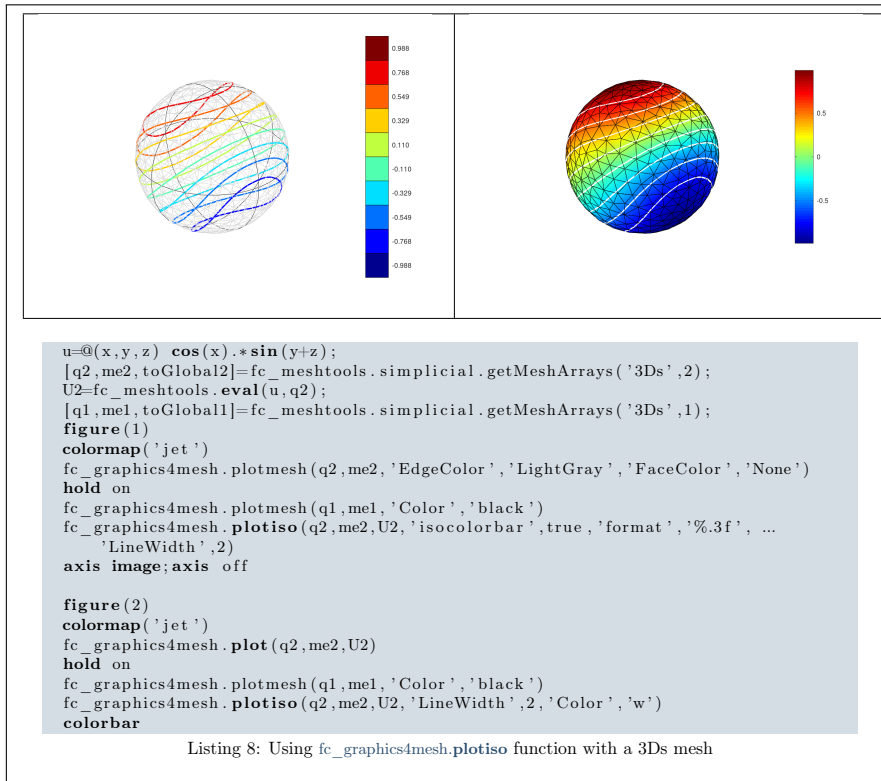
figure(2)
fc_graphics4mesh.plot(q2,me2,U2,'plane',true)
hold on
fc_graphics4mesh.plotiso(q2,me2,U2,'plane',true,...
'Color','w','isorange',0,'LineWidth',2)
fc_graphics4mesh.plotmesh(q1,me1,'LineWidth',2,'Color','k')
colorbar
shading interp
axis image;axis off

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

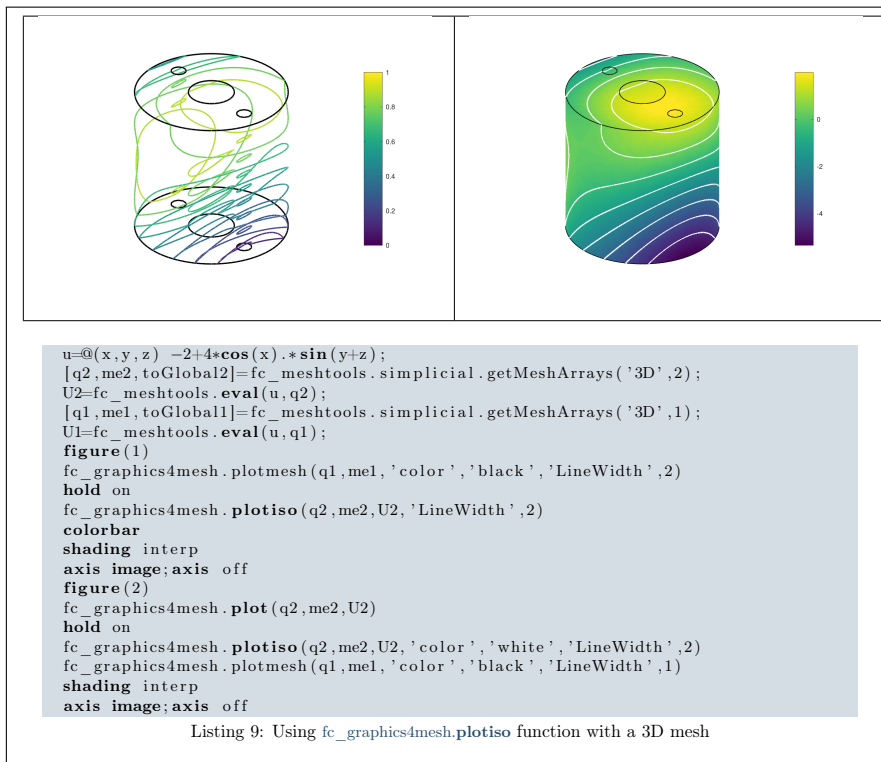
```

Listing 7: Using `fc_graphics4mesh.plotiso` function with a 2D mesh

**3Ds example :** the following code is part of the `fc_graphics4mesh.demos.plotiso3Ds` function.



**3D example :** the following code is part of the `fc_graphics4mesh.demos.plotiso3D` function.



## 7 `fc_graphics4mesh.slicemesh` function

The `fc_graphics4mesh.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` package. 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 used a  $n$ -by-4 array  $P$  where each line define a plane.

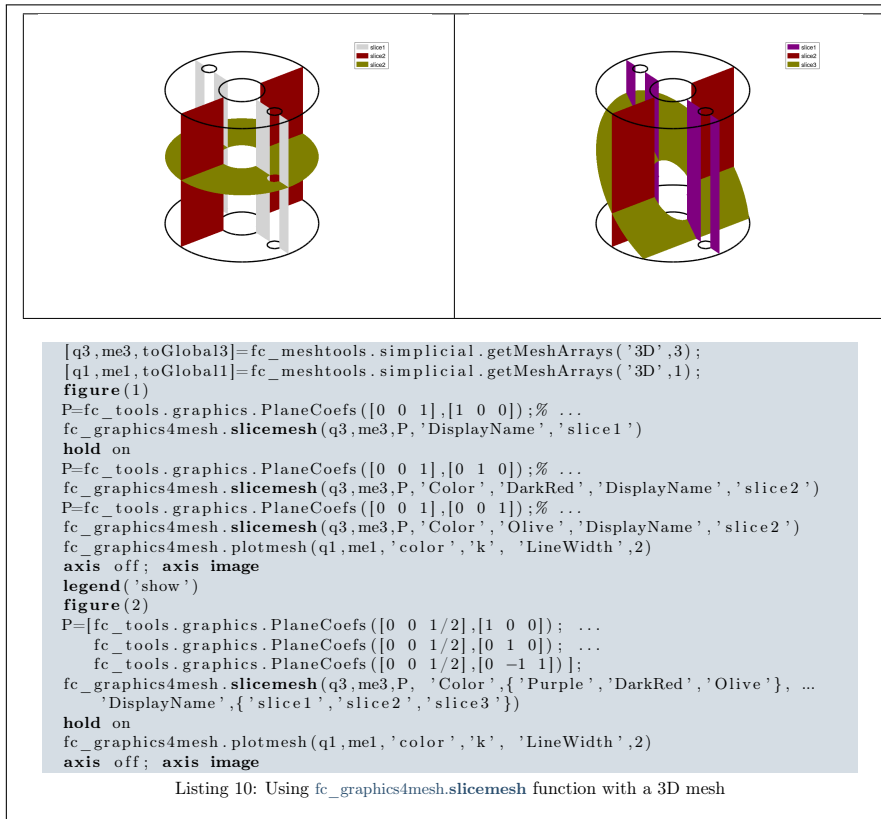
`slicemesh(q,me,P,Name,Value,...)` specifies function options using one or more `Name,Value` pair arguments. Options of first level are



- '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 :** the following code is part of the `fc_graphics4mesh.demos.slicemesh3D` function.



## 8 `fc_graphics4mesh.slice` function

The `fc_graphics4mesh.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,...)

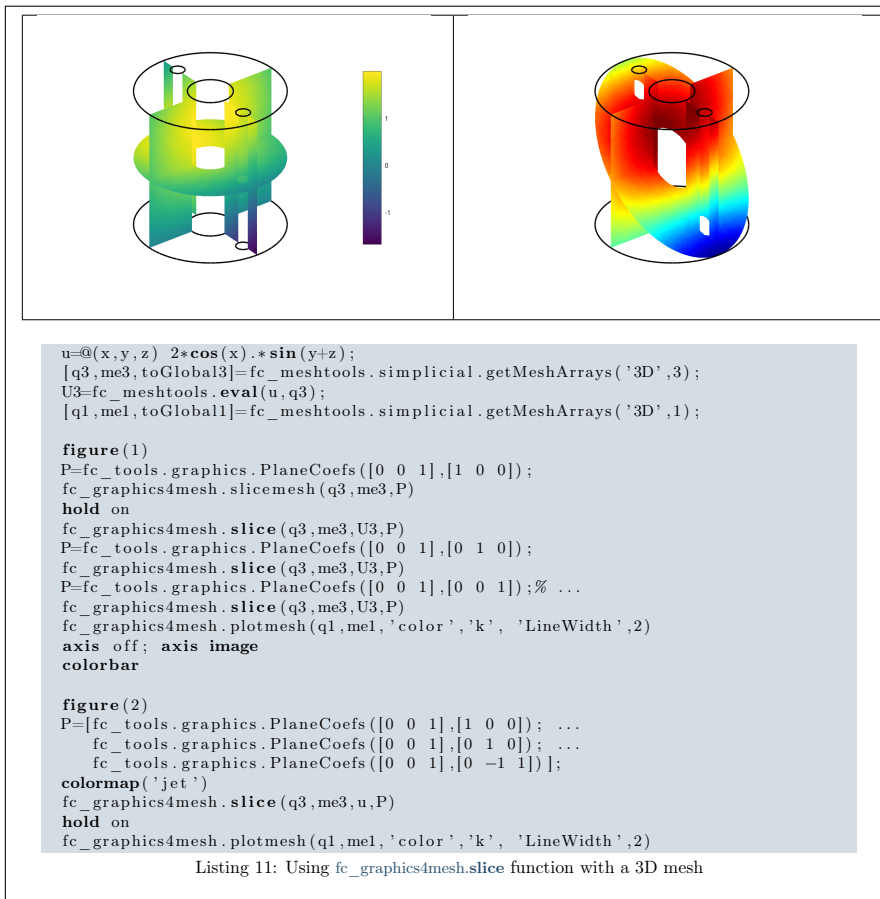
```

### 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** package. 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 used 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 :** the following code is part of the `fc_graphics4mesh.demos.slice3D` function.



## 9 `fc_graphics4mesh.sliceiso` function

The `fc_graphics4mesh.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** package. The 1-by-4 array  $P$ , is obtained with  $P=fc\_tools.graphicsPlaneCoefs(Q,V)$  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

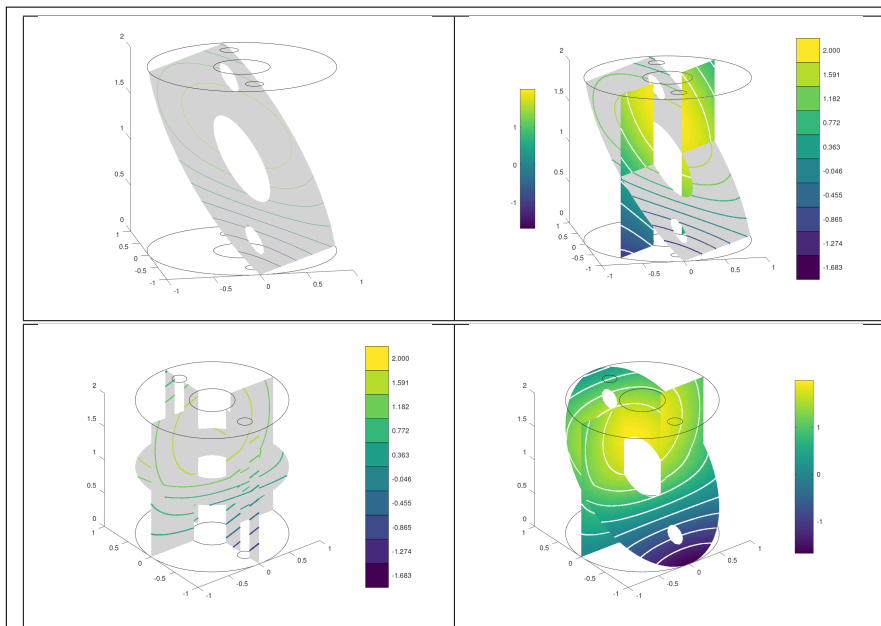
- `'niso'` : to specify the number of isolines (default : 10)
- `'isorange'` : to specify the list of isovalues (default : empty)
- `'color'` : to specify one color for all isolines (default : empty)
- `'isocolorbar'` : if true display a colorbar.Default is false.
- `'format'` : to specify the format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

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 code is part of the `fc_graphics4mesh.demos.sliceiso3D` function.



```

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
fc_graphics4mesh.plotmesh(q1,me1,'color','k')
fc_graphics4mesh.sliceiso(q3,me3,U3,P)
axis equal;axis image
view(-11,15)

```

```

figure(2)
P=fc_tools.graphics.PlaneCoefs([0 0 1],[1 -1 1]);
fc_graphics4mesh.slicemesh(q3,me3,P)
hold on
fc_graphics4mesh.plotmesh(q1,me1,'color','k')
fc_graphics4mesh.sliceiso(q3,me3,U3,P,'Linewidth',2, ...
    'isocolorbar',true,'LineWidth',2,'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','LineWidth',2);
axis equal;axis image
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
fc_graphics4mesh.plotmesh(q1,me1,'color','k')
fc_graphics4mesh.sliceiso(q3,me3,U3,P,'isocolorbar',true,'LineWidth',2, ...
    'format','%3f');
axis equal;axis image

```

```

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
fc_graphics4mesh.plotmesh(q1,me1,'color','k')
fc_graphics4mesh.sliceiso(q3,me3,U3,P,'Color','w','LineWidth',2);
caxis([min(U3),max(U3)])
axis equal;axis image

```

Listing 12: Using `fc_graphics4mesh.sliceiso` function with a 3D mesh

## 10 `fc_graphics4mesh.plotquiver` function

The function `fc_graphics4mesh.plotquiver` displays vector field data 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 a 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

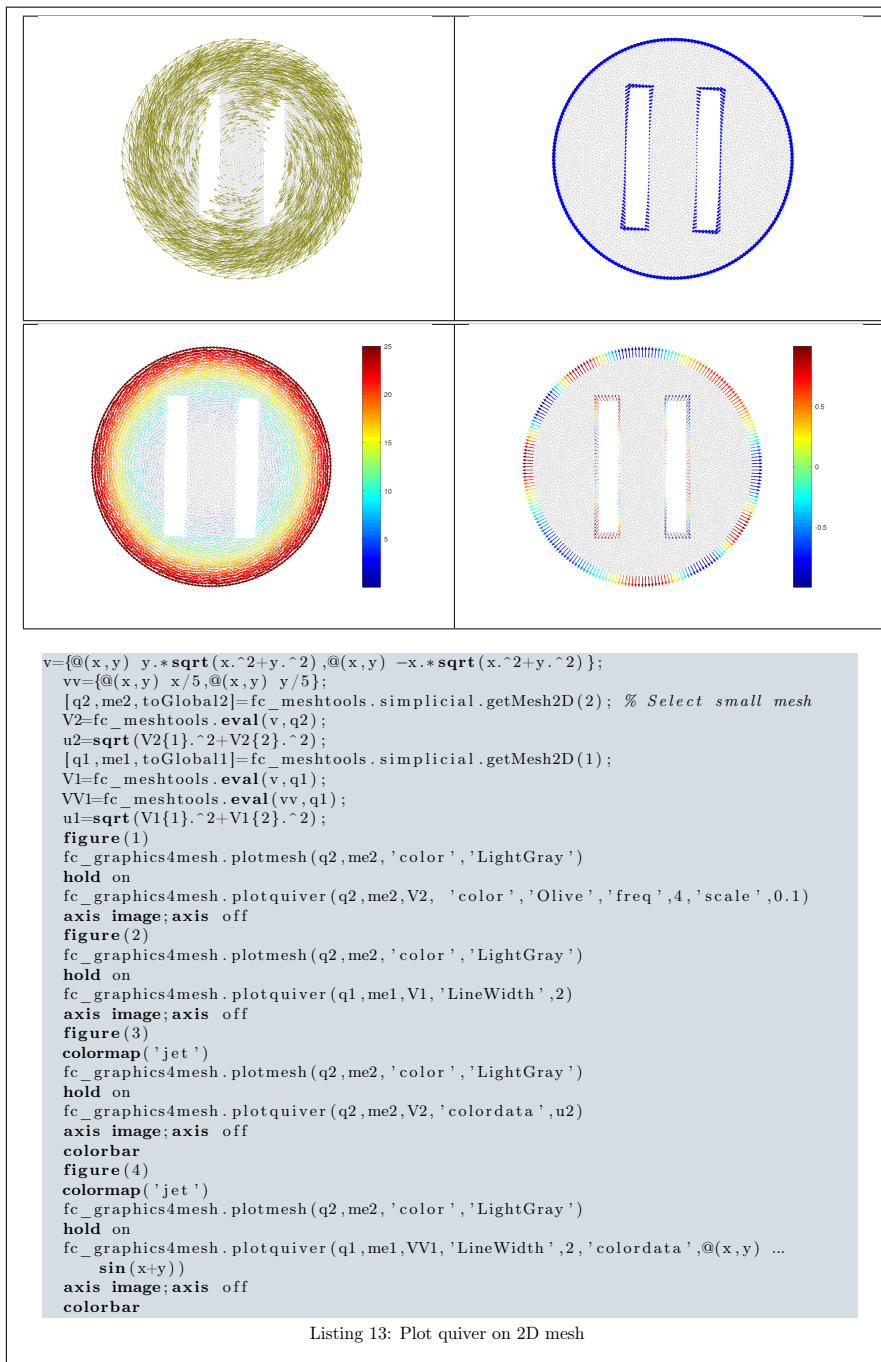
- `'freq'` : quiver frequency, (default : 1)
- `'scale'` : quiver scale, (default is `fc_meshtools.getCharacteristicLength(q)/20`)
- `'color'` : set one color for all quivers (default: default color of the **quiver** or **quiver3** functions). Cannot be used with `'colordata'` option.
- `'colordata'`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

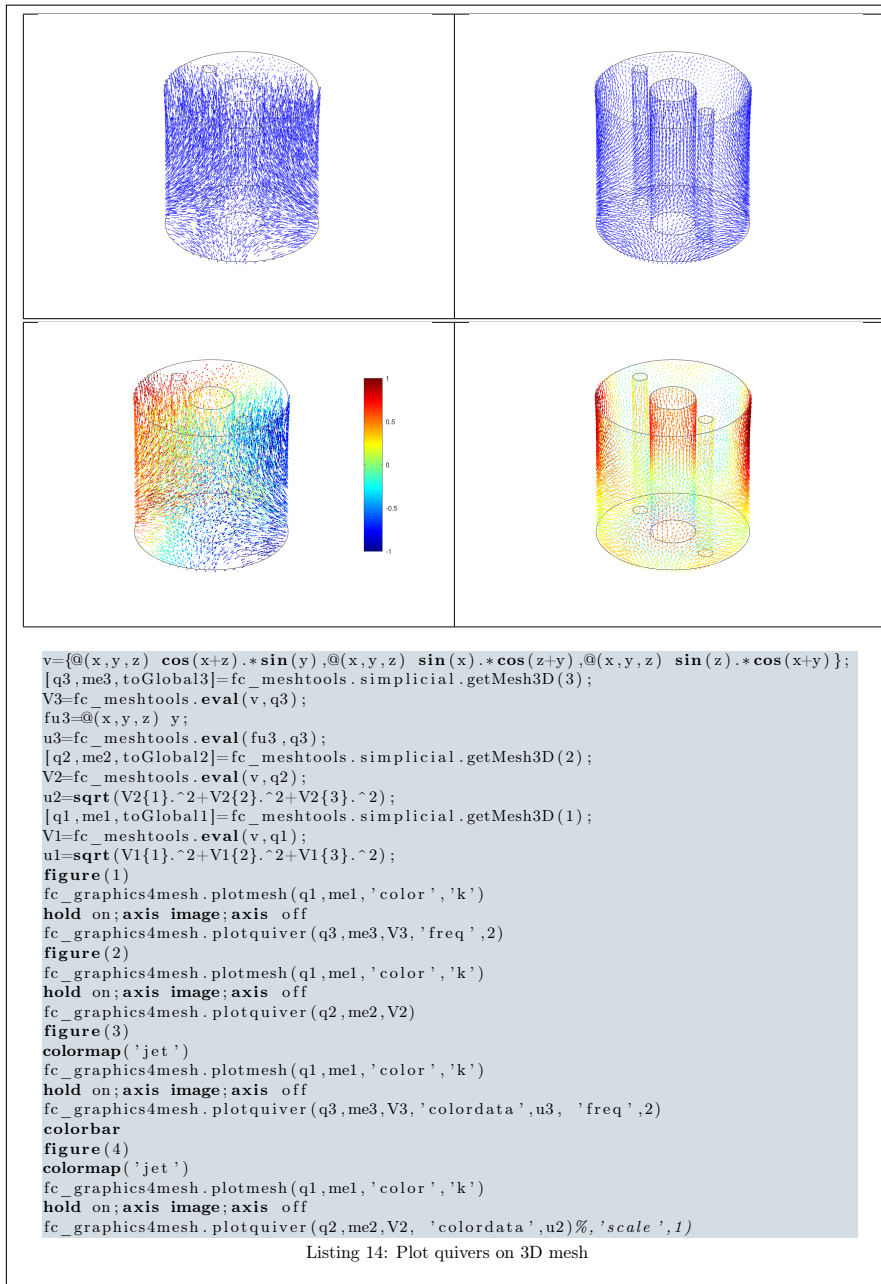
- In dimension 3 and with empty `'colordata'`, the **quiver3** function is used.
- In dimension 2 and with empty `'colordata'`, the **quiver** function is used.
- In dimension 2 or 3 and with no empty `'colordata'`, the third party `fc_tools.graphics.vfield3.vfield3` function is used.

### 2D example

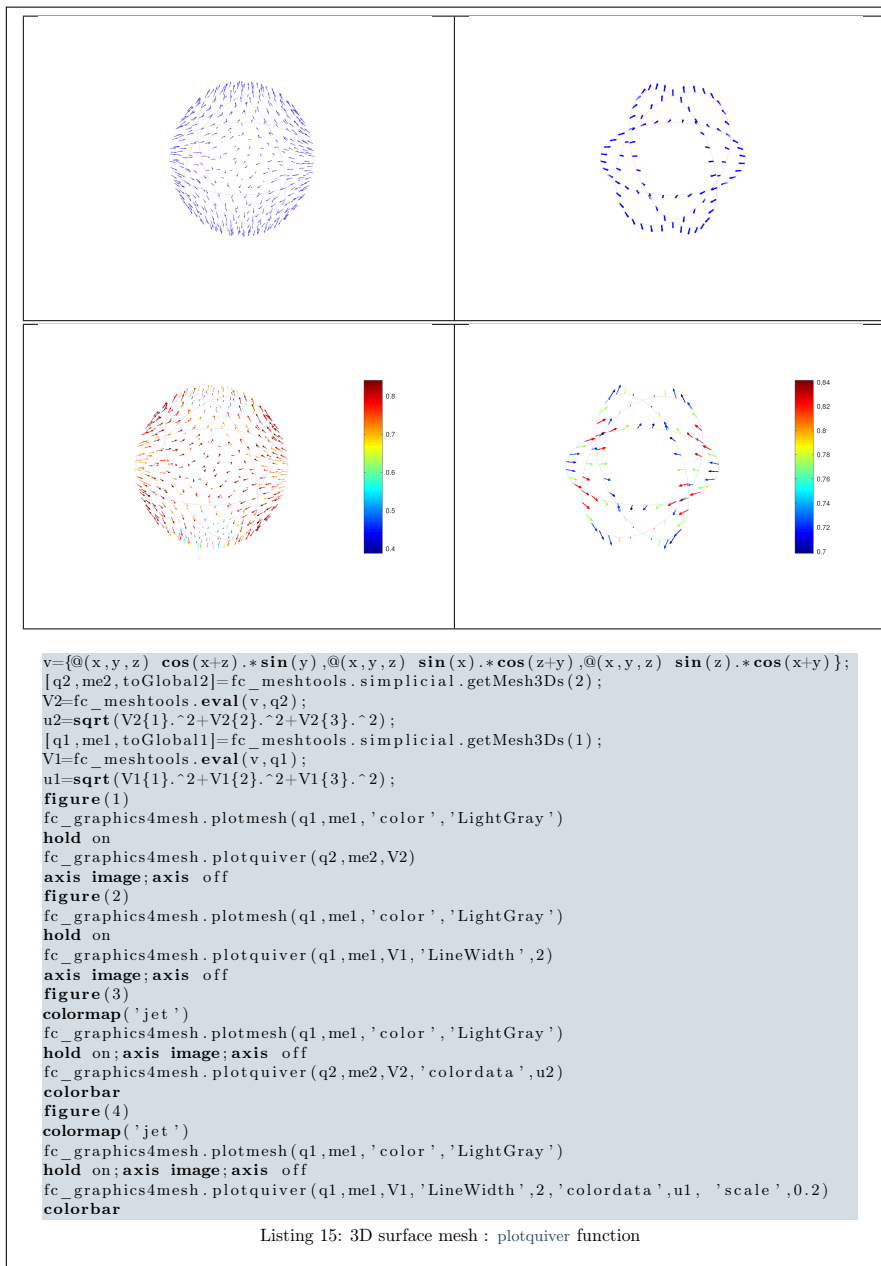


Listing 13: Plot quiver on 2D mesh

### 3D example



### 3D surface example



## 11 fc\_graphics4mesh.plotnodes function

The function `fc_graphics4mesh.plotnodes` displays the nodes of a given mesh nodes array

**Syntaxe**



```
fc_graphics4mesh.plotnodes(q)
fc_graphics4mesh.plotnodes(q,Name,Value, ...)
```

## Description

`plotnodes(q)` displays all the nodes of the array `q` with a specific marker.

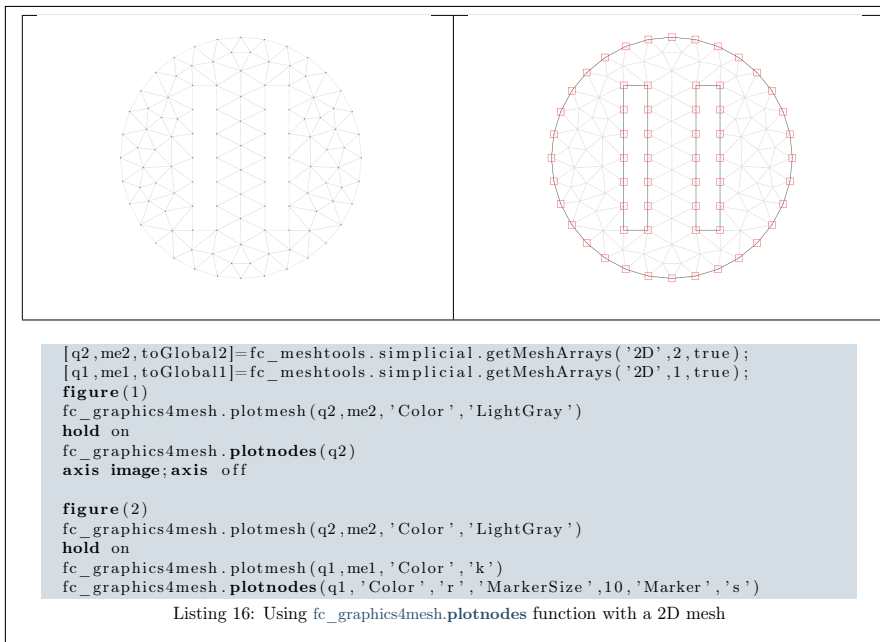
`plotnodes(q,Name,Value, ...)` specifies options using one or more `Name,Value` pair arguments. Options of first level are

- 'Marker' : to specify the marker (default : '.'),
- 'MarkerSize' : to specify the marker size (default : 6),
- 'idx' : to specify indices of the nodes to be displayed,

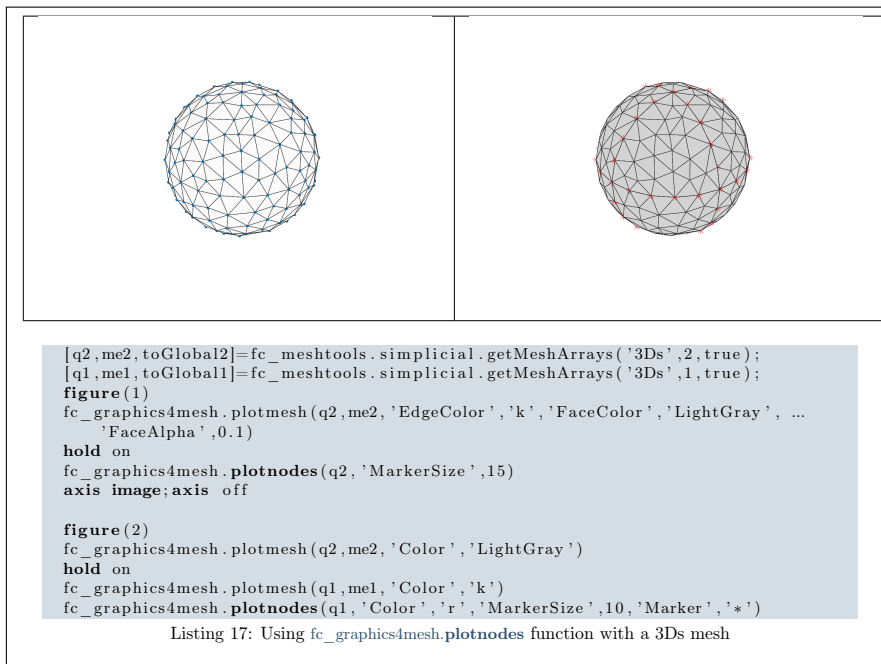
The options of second level depend on the dimension :

- if dimension 2, then options are those of the `plot` function,
- if dimension 3, then options are those of the `plot3` function.

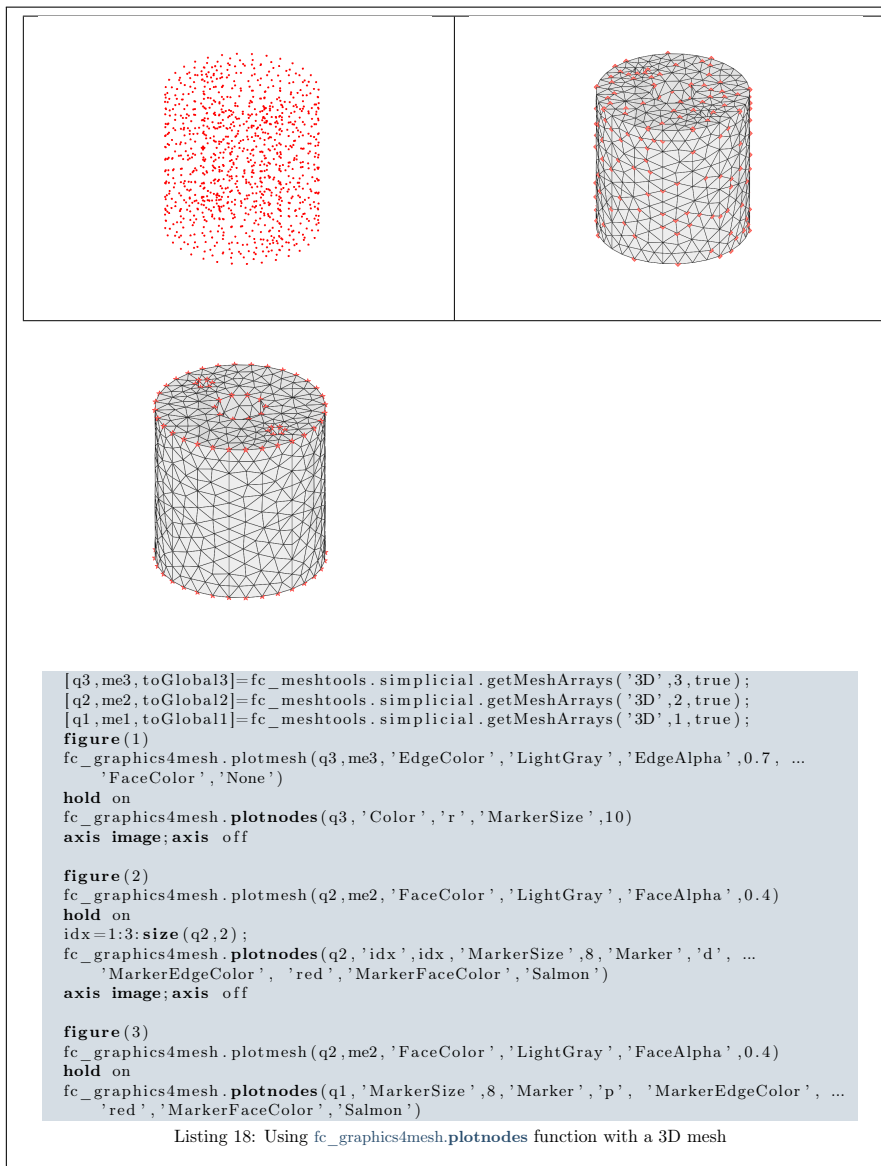
**2D example :** the following code is part of the `fc_graphics4mesh.demos.plotnodes2D` function.



**3Ds example :** the following code is part of the `fc_graphics4mesh.demos.plotnodes3Ds` function.



**3D example :** the following code is part of the `fc_graphics4mesh.demos.plotnodes3D` function.



## 12 fc\_graphics4mesh.plotnodesidx function

The function `fc_graphics4mesh.plotnodesidx` displays indices of the given mesh nodes array

### Syntaxe

```

fc_graphics4mesh.plotnodesidx(q,
fc_graphics4mesh.plotnodesidx(q,Name,Value,...)

```

### Description

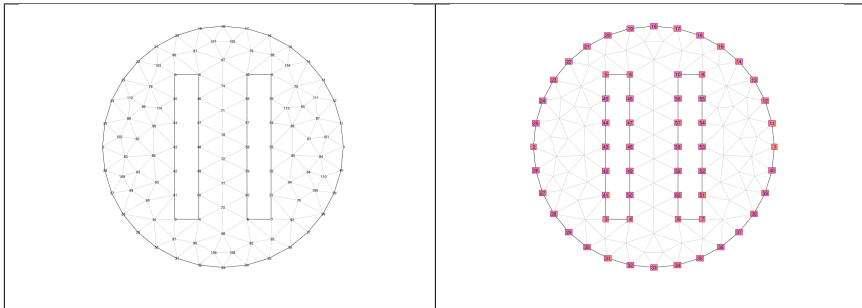
`plotnodesidx(q)` displays all the numbers/indices of the nodes array `q`

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

- 'toGlobal' : to specify other indices to display,
- 'idx' : to select particular indices.

The options of second level are those of the `text` function.

**2D example :** the following code is part of the `fc_graphics4mesh.demos.plotnodesidx2D` function.

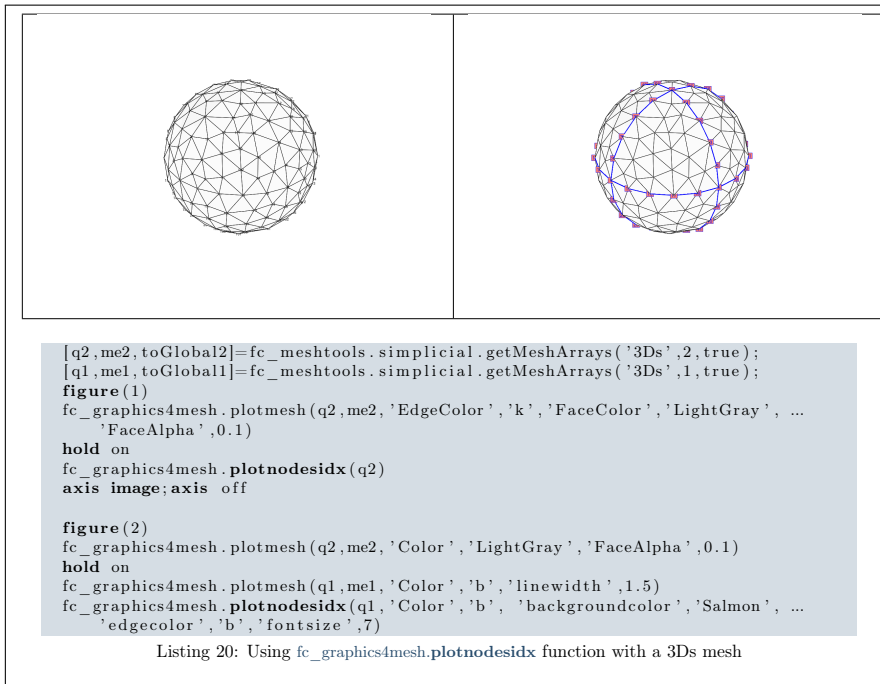


```
[q2,me2,toGlobal2]=fc_meshtools.simplicial.getMeshArrays('2D',2,true);
[q1,me1,toGlobal1]=fc_meshtools.simplicial.getMeshArrays('2D',1,true);
figure(1)
fc_graphics4mesh.plotmesh(q2,me2,'Color','LightGray')
hold on
fc_graphics4mesh.plotmesh(q1,me1,'Color','k')
fc_graphics4mesh.plotnodesidx(q2)
axis image,axis off

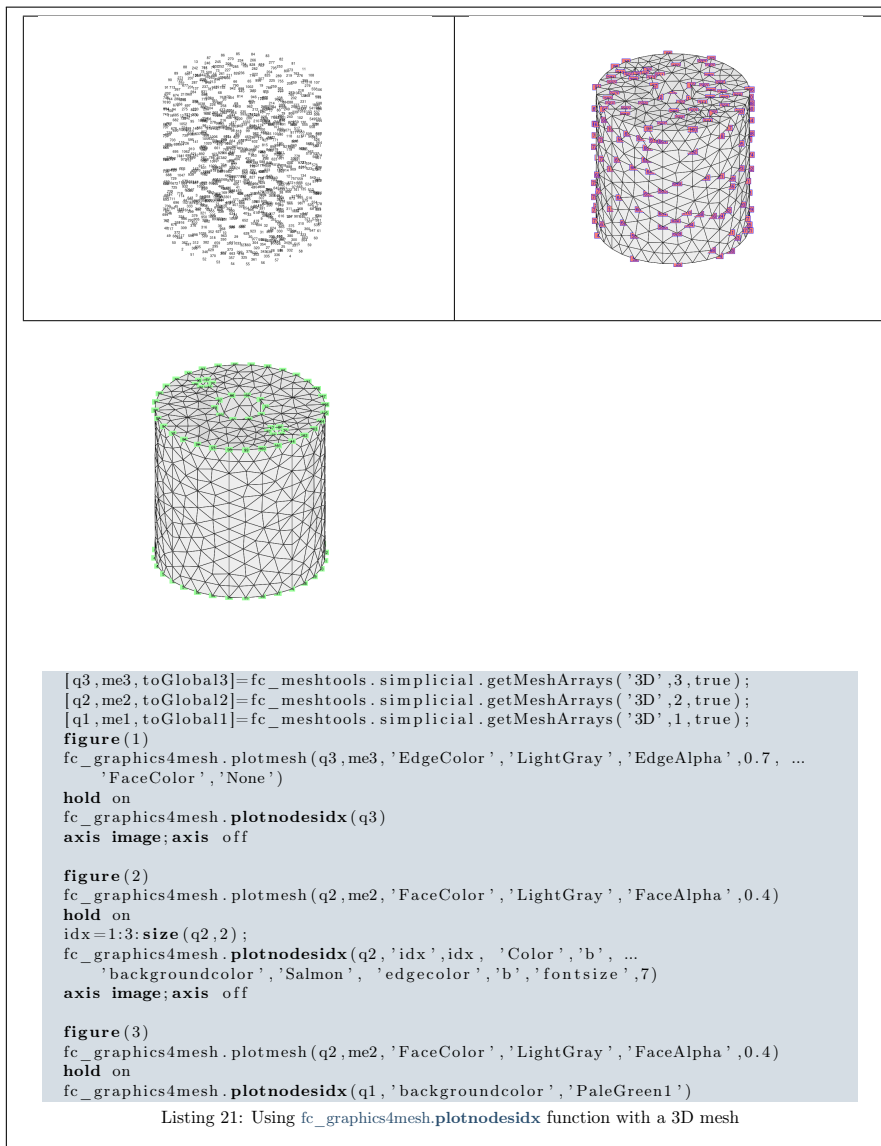
figure(2)
fc_graphics4mesh.plotmesh(q2,me2,'Color','LightGray')
hold on
fc_graphics4mesh.plotmesh(q1,me1,'Color','k')
fc_graphics4mesh.plotnodesidx(q1,'Color','b','backgroundcolor','Salmon', ...
'edgecolor','purple','fontSize',7)
```

Listing 19: Using `fc_graphics4mesh.plotnodesidx` function with a 2D mesh

**3Ds example :** the following code is part of the `fc_graphics4mesh.demos.plotnodesidx3Ds` function.



**3D example :** the following code is part of the `fc_graphics4mesh.demos.plotnodesidx3D` function.



## 13 `fc_graphics4mesh.plotelementsidx` function

The function `fc_graphics4mesh.plotelementsidx` displays indices of a given mesh connectivity array

### Syntaxe

```

fc_graphics4mesh.plotelementsidx(q,me)
fc_graphics4mesh.plotelementsidx(q,me,Name,Value, ...
... )

```

## Description

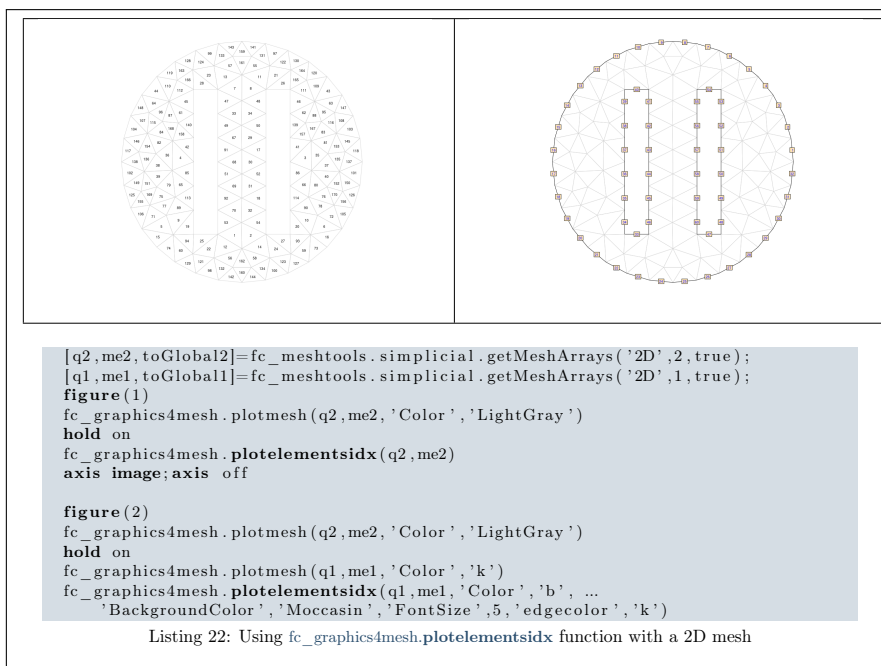
`plotelementsidx(q,me)` displays all the numbers/indices of the connectivity array `me`.

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

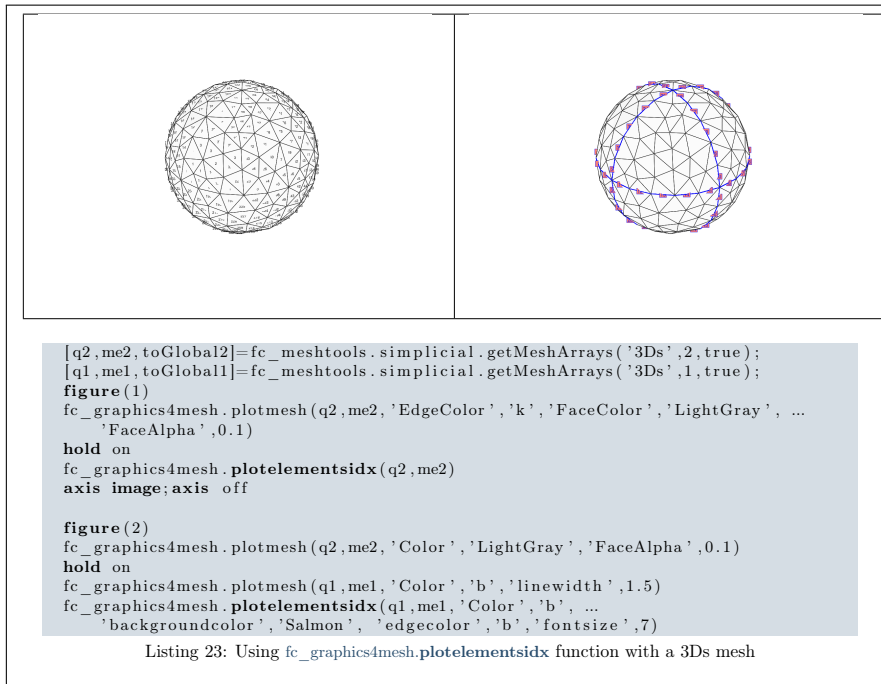
- `'toGlobal'` : to specify other indices to display,
- `'idx'` : to select particular indices.

The options of second level are those of the `text` function.

**2D example :** the following code is part of the `fc_graphics4mesh.demos.plotelementsidx2D` function.

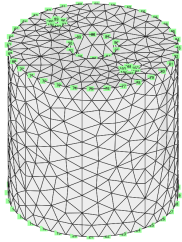
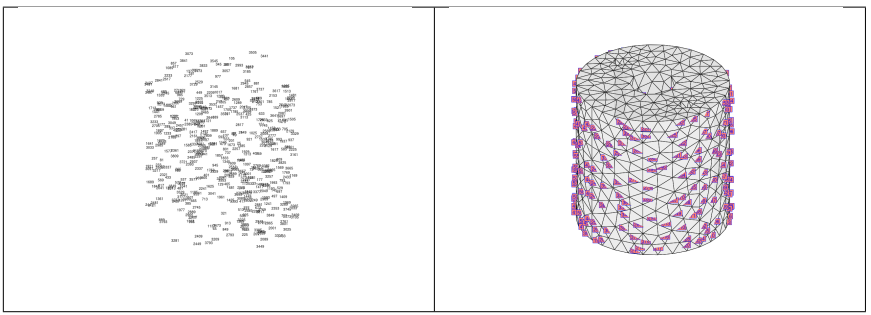


**3Ds example :** the following code is part of the `fc_graphics4mesh.demos.plotelementsidx3Ds` function.



**3D example :** the following code is part of the `fc_graphics4mesh.demos.plotelementsidx3D` function.





```
[q3,me3,toGlobal3]=fc_meshtools.simplicial.getMeshArrays('3D',3,true);
[q2,me2,toGlobal2]=fc_meshtools.simplicial.getMeshArrays('3D',2,true);
[q1,me1,toGlobal1]=fc_meshtools.simplicial.getMeshArrays('3D',1,true);
figure(1)
fc_graphics4mesh.plotmesh(q3,me3,'EdgeColor','LightGray','EdgeAlpha',0.7, ...
    'FaceColor','None')
hold on
idx=1:8: size(me3,2);
fc_graphics4mesh.plotelementsidx(q3,me3,'idx',idx)
axis image;axis off

figure(2)
fc_graphics4mesh.plotmesh(q2,me2,'FaceColor','LightGray','FaceAlpha',0.4)
hold on
idx=1:3: size(q2,2);
fc_graphics4mesh.plotelementsidx(q2,me2,'idx',idx, 'Color','b', ...
    'backgroundcolor','Salmon', 'edgecolor','b','fontsize',7)
axis image;axis off

figure(3)
fc_graphics4mesh.plotmesh(q2,me2,'FaceColor','LightGray','FaceAlpha',0.4)
hold on
fc_graphics4mesh.plotelementsidx(q1,me1,'backgroundcolor','PaleGreen1')
```

Listing 24: Using fc\_graphics4mesh.plotelementsidx function with a 3D mesh

## Informations for git maintainers of the Octave package

git informations on the packages used to build this manual

```
-----  
name : fc-graphics4mesh  
tag : 0.1.2  
commit : e922041e090275b69207ed6d4fb52df06f5e3fba  
date : 2020-03-07  
time : 16-09-48  
status : 0  
-----  
name : fc-tools  
tag : 0.0.31  
commit : 3a25f08630ff69e7f8ef4ab451671e454c492e36  
date : 2020-02-23  
time : 11-27-44  
status : 0  
-----  
name : fc-bench  
tag : 0.1.2  
commit : 666dc60d1277f5fa9c99dee4ae1c33270f22c57d  
date : 2020-02-16  
time : 06-38-46  
status : 0  
-----  
name : fc-amat  
tag : 0.1.2  
commit : 957340f6e71d805dbd8b9d04c434b24fd3f92591  
date : 2020-02-16  
time : 06-39-42  
status : 0  
-----  
name : fc-meshtools  
tag : 0.1.3  
commit : cdbc41bc98af4e4facc1746024aced1f21aae53  
date : 2020-02-17  
time : 10-52-56  
status : 0  
-----
```

git informations on the L<sup>A</sup>T<sub>E</sub>X package used to build this manual

```
-----  
name : fctools  
tag :  
commit : 57968c4a96c2593cccc9da9efd3e52b2ff012cb5  
date : 2020-02-07  
time : 06:41:09  
status : 1  
-----
```

Using the remote configuration repository:

```
url      ssh://lagagit/MCS/Cuvelier/Matlab/fc-config  
commit   5e45f254c19250ff5077a8477825e6c975245689
```