



Octave package, User's Guide* version 0.1.6

François Cuvelier[†]

March 4, 2023

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

1	Introduction	2
2	Installation	2
3	Mesh	2
4	<code>fc_graphics4mesh.plotmesh</code> function	2
5	<code>fc_graphics4mesh.plot</code> function	7
6	<code>fc_graphics4mesh.plotiso</code> function	9
7	<code>fc_graphics4mesh.slicemesh</code> function	13
8	<code>fc_graphics4mesh.slice</code> function	14
9	<code>fc_graphics4mesh.sliceiso</code> function	15
10	<code>fc_graphics4mesh.plotquiver</code> function	17
11	<code>fc_graphics4mesh.scatter</code> function	20
12	<code>fc_graphics4mesh.plotnodes</code> function	23
13	<code>fc_graphics4mesh.plotnodesidx</code> function	26

* $\LaTeX$ manual, revision 0.1.6.a, compiled with Octave 7.3.0, and packages `fc-graphics4mesh`[0.1.6], `fc-tools`[0.0.35], `fc-bench`[0.1.3], `fc-amat`[0.1.3], `fc-meshtools`[0.1.4]

[†]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.

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.

2 Installation

to do ...

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_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 T_h -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 : `[]` (no cut).

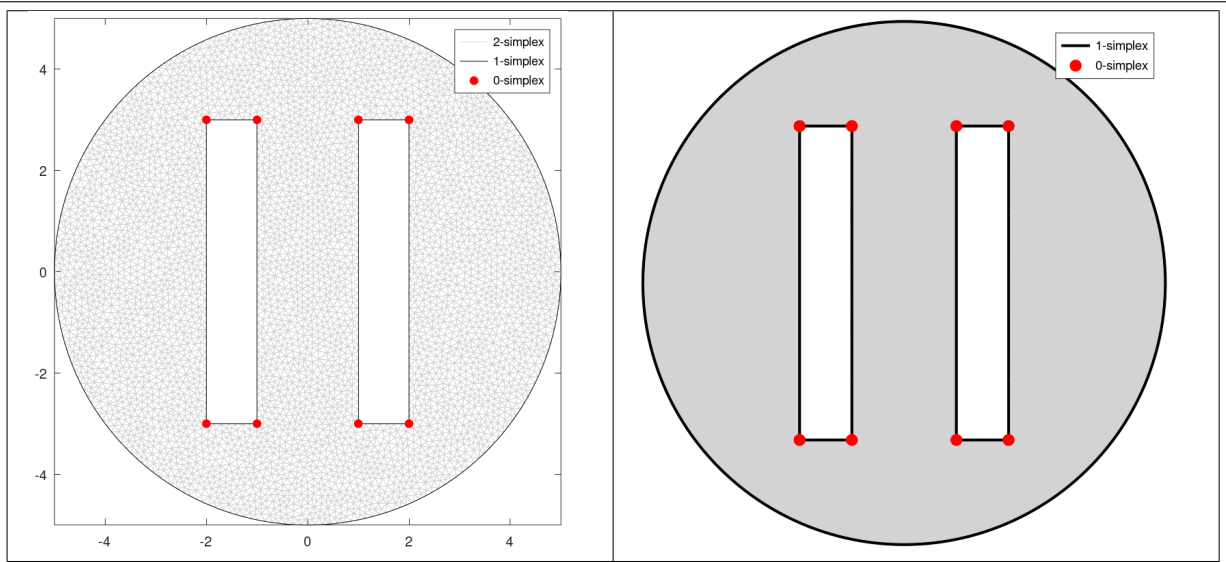
- `'inLegend'` : to add this mesh in a legend if `true`. Default is `false`.
- `'DisplayName'` : to specify the name used in a legend for this mesh. Then the (`'inLegend'` option is forced to `true`).

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.

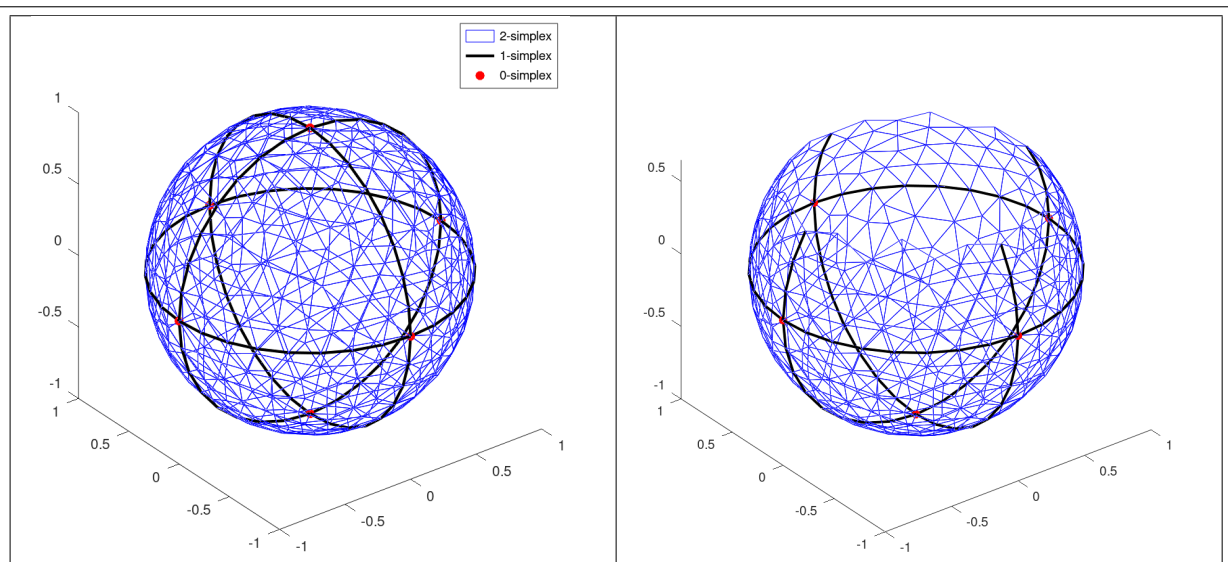


```
[q2,me2,toGlobal2]=fc_meshtools.simplicial.getMesh2D(2);
[q1,me1,toGlobal1]=fc_meshtools.simplicial.getMesh2D(1);
[q0,me0,toGlobal0]=fc_meshtools.simplicial.getMesh2D(0);
figure(1)
fc_graphics4mesh.plotmesh(q2,me2,'Color','LightGray','DisplayName','2-simplex')
hold on
fc_graphics4mesh.plotmesh(q1,me1,'color','black','DisplayName','1-simplex')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

figure(2)
fc_graphics4mesh.plotmesh(q2,me2,'fill',true,'Color','LightGray')
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
```

Listing 1: Using `fc_graphics4mesh.plotmesh` function with a 2D mesh

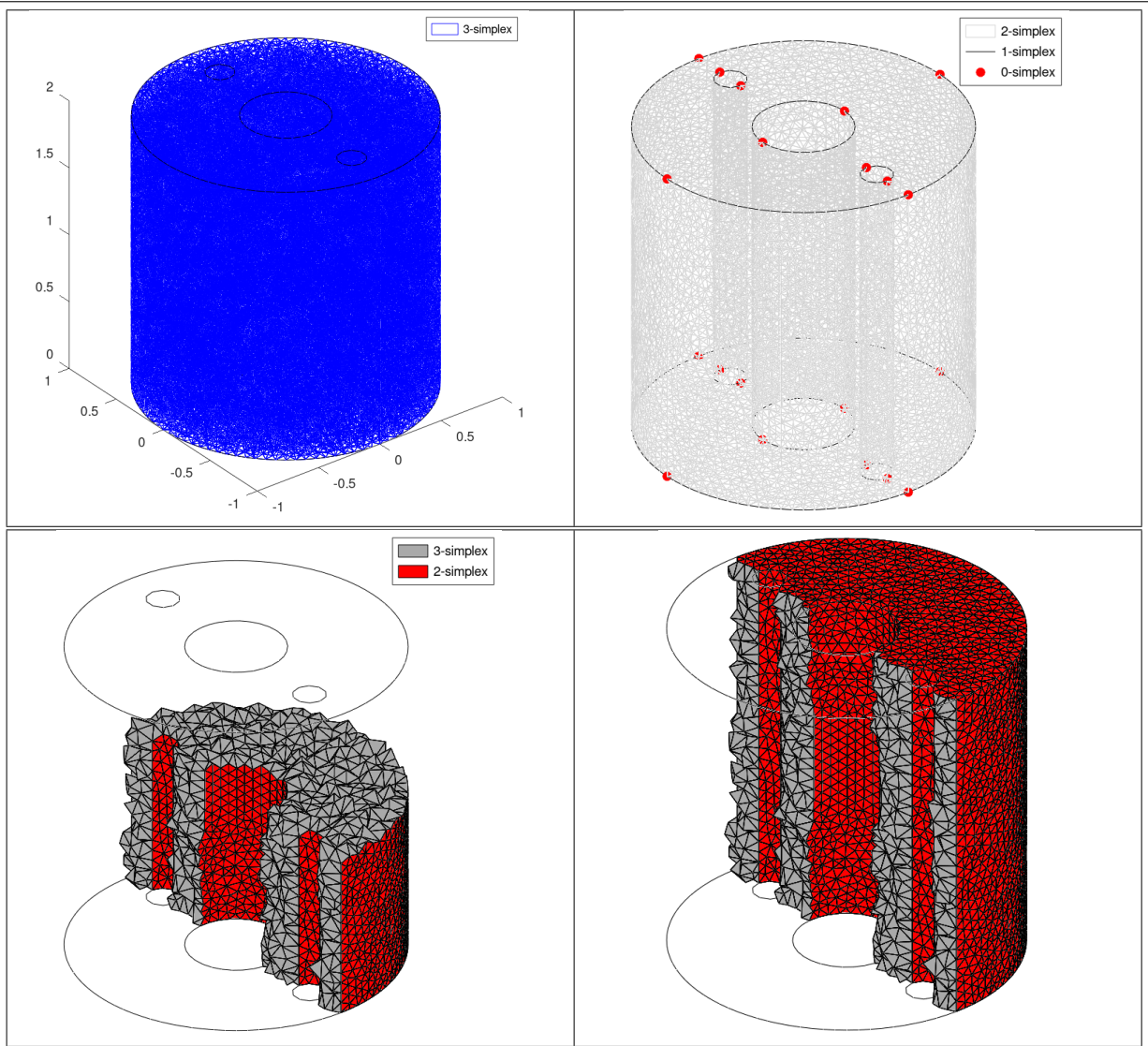
3Ds example : the following code is part of the `fc_graphics4mesh.demos.plotmesh3Ds` function.



```
[q2,me2,toGlobal2]=fc_meshtools.simplicial.getMesh3Ds(2);
[q1,me1,toGlobal1]=fc_meshtools.simplicial.getMesh3Ds(1);
[q0,me0,toGlobal0]=fc_meshtools.simplicial.getMesh3Ds(0);
figure(1)
fc_graphics4mesh.plotmesh(q2,me2,'EdgeColor','blue','FaceColor','None','DisplayName','2-simplex')
hold on
fc_graphics4mesh.plotmesh(q1,me1,'color','black','Linewidth',2,'DisplayName','1-simplex')
fc_graphics4mesh.plotmesh(q0,me0,'color','red','DisplayName','0-simplex')
view(3);axis image
fc_graphics4mesh.legend()
P=fc_tools.graphics.PlaneCoefs([0 0 1/2],[0 0 1]);
figure(2)
fc_graphics4mesh.plotmesh(q2,me2,'EdgeColor','blue','FaceColor','None','cutPlane',P)
hold on
fc_graphics4mesh.plotmesh(q1,me1,'color','black','Linewidth',2,'cutPlane',P)
fc_graphics4mesh.plotmesh(q0,me0,'color','red','cutPlane',P)
```

Listing 2: Using `fc_graphics4mesh.plotmesh` function with a 3Ds mesh

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
fc_graphics4mesh.legend()

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
fc_graphics4mesh.legend()

P=[fc_tools.graphics.PlaneCoefs([0 0 1],[0 0 1]); ...
fc_tools.graphics.PlaneCoefs([0 0 1],[-1 0 0])];
figure(3)
view(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
fc_graphics4mesh.legend()

P=fc_tools.graphics.PlaneCoefs([0 0 1],[-1 0 0]);
figure(4)
view(3)
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

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

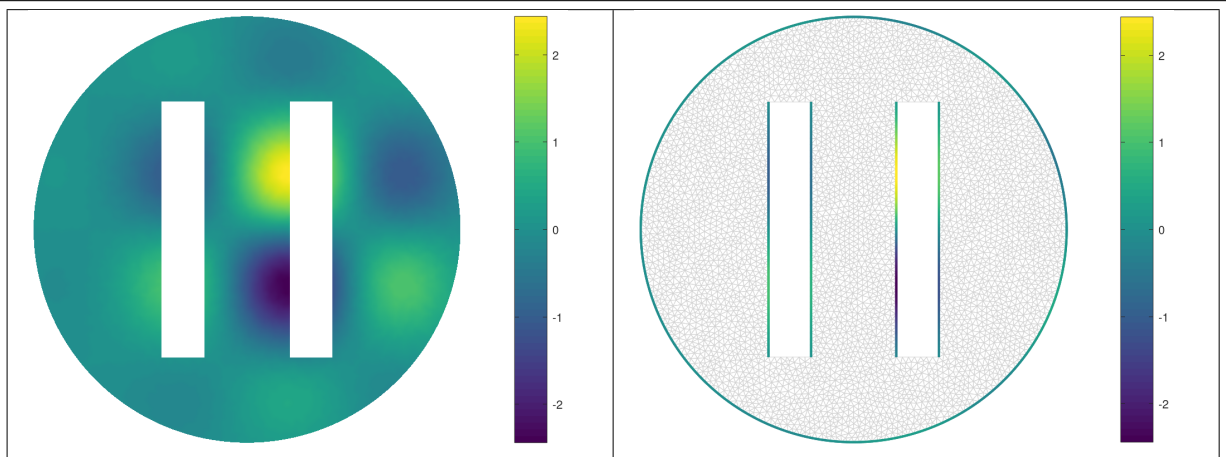
The normal vector $P(i,1:3)$ pointed to the part of the mesh not displayed. default : `[]` (no cut).

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.

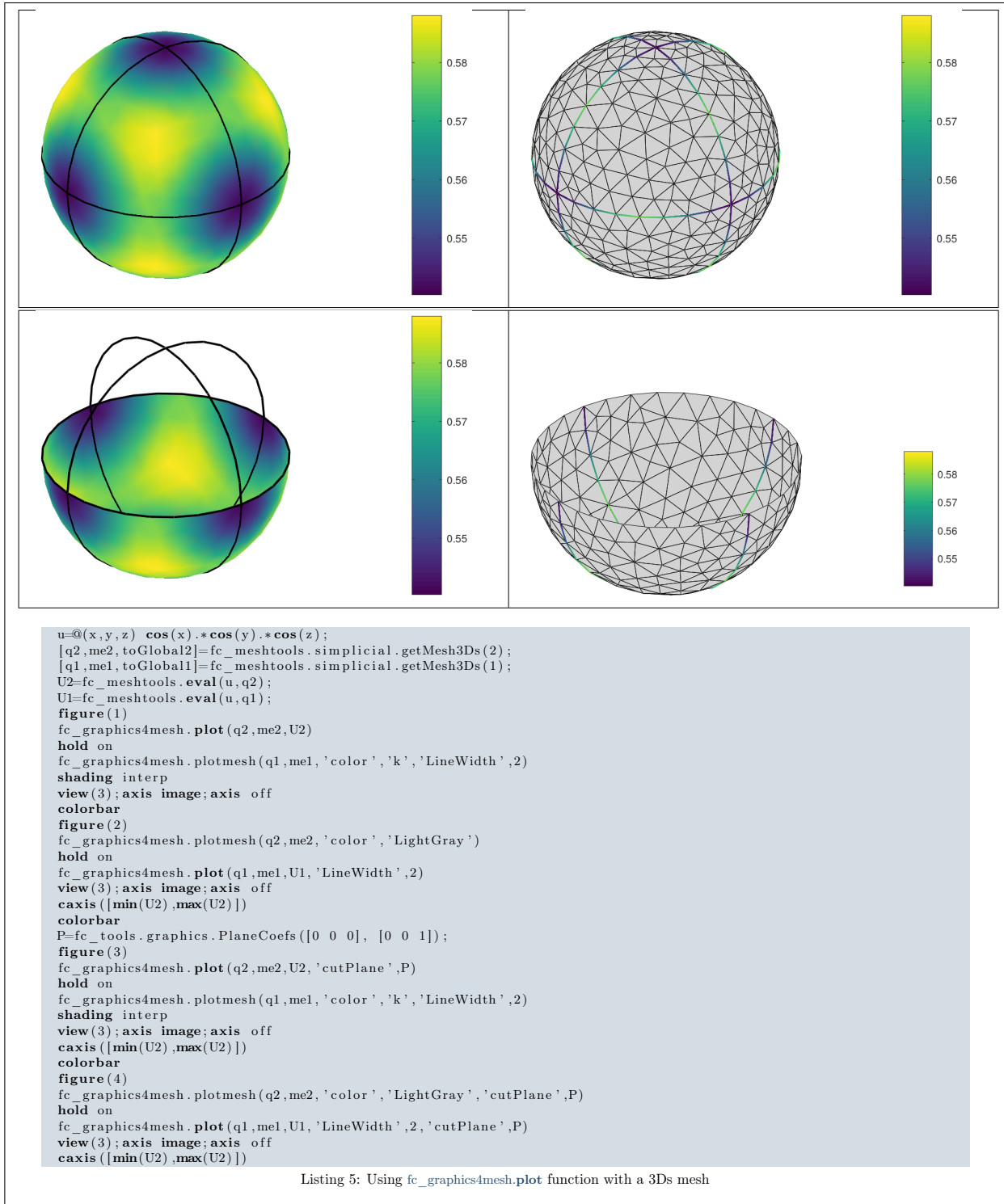


```
u=@(x,y) 3*exp(-(x-1).^2+y.^2)/10).*cos(x-1).*sin(y);
[q2,me2,toGlobal2]=fc_meshtools.simplicial.getMeshArrays('2D',2);
[q1,me1,toGlobal1]=fc_meshtools.simplicial.getMeshArrays('2D',1);
U2=fc_meshtools.eval(u,q2);
figure(1)
fc_graphics4mesh.plot(q2,me2,U2,'plane',true)
shading interp
axis image;axis off
colorbar

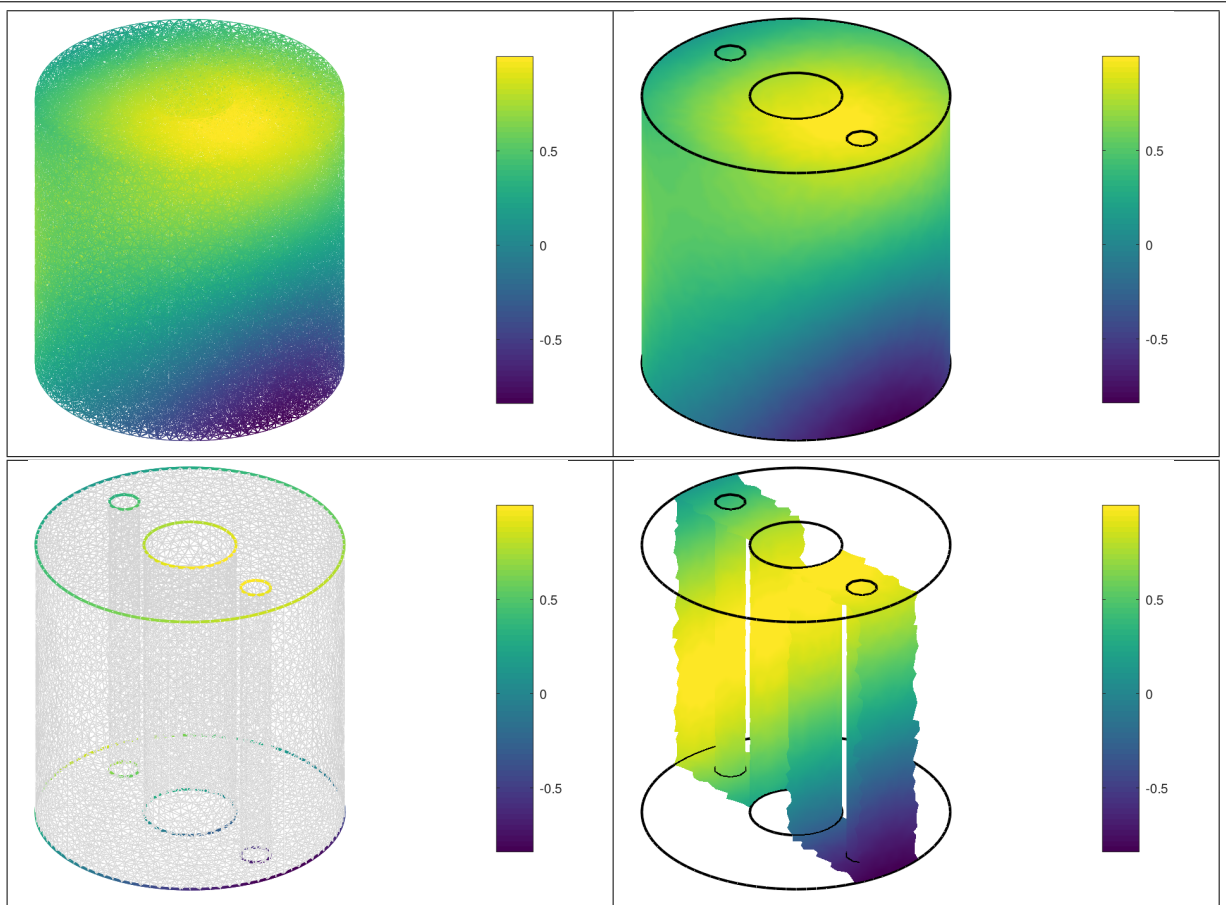
figure(2)
fc_graphics4mesh.plotmesh(q2,me2,'color','LightGray')
hold on
fc_graphics4mesh.plot(q1,me1,U2(toGlobal1),'LineWidth',2,'Plane',true)
caxis([min(U2) max(U2)])
axis image;axis off
```

Listing 4: Using `fc_graphics4mesh.plot` function with a 2D mesh

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
view(3);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
view(3);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)
view(3);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
view(3);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)

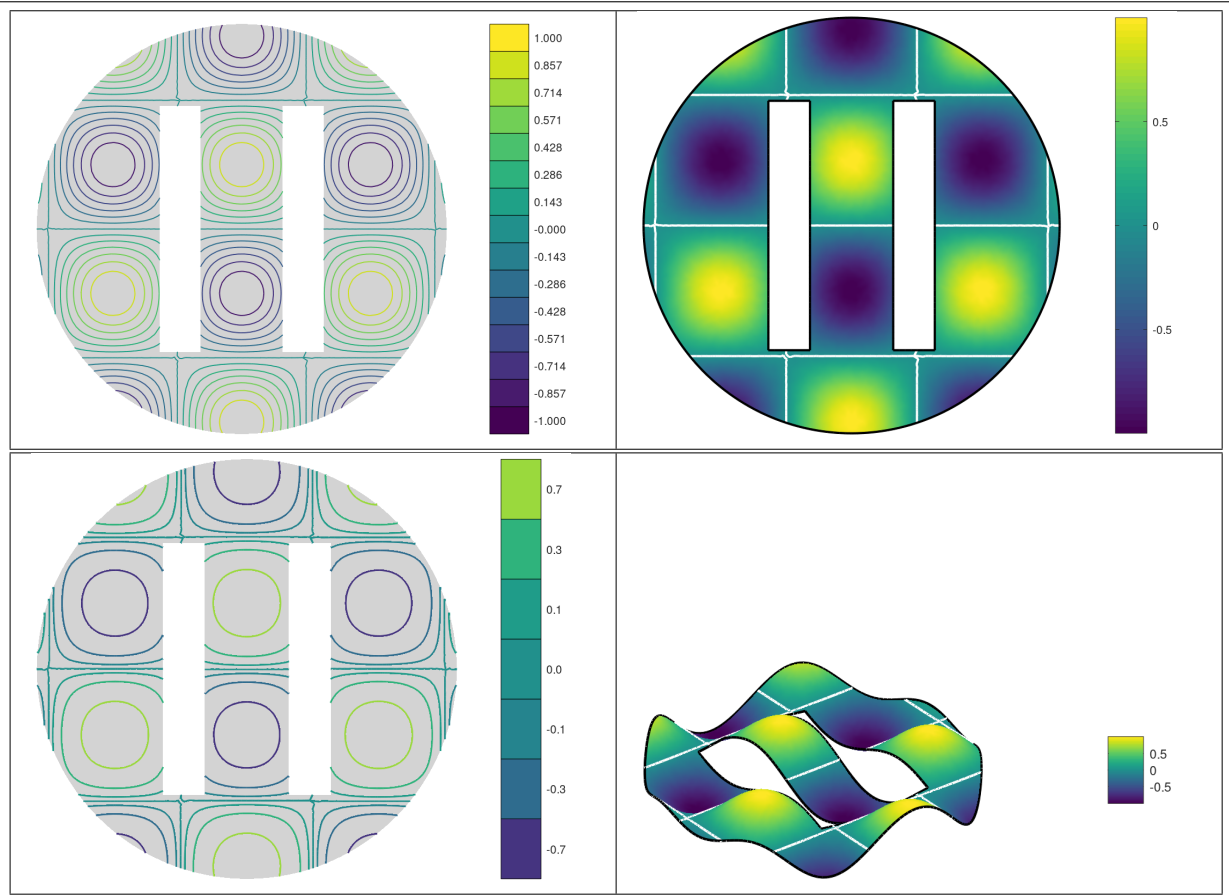
The options of second level are all options of

- `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

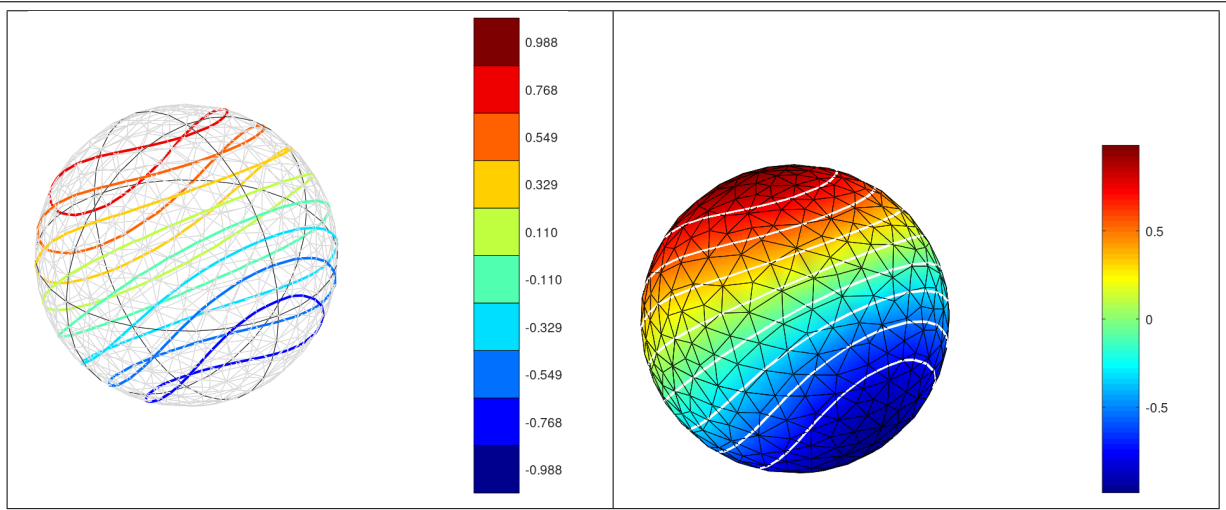
figure(2)
fc_graphics4mesh.plot(q2,me2,U2,'plane',true)
hold on
fc_graphics4mesh.plotiso(q2,me2,U2,'plane',true,'Color','w','isorange',0,'LineWidth',2)
fc_graphics4mesh.plotmesh(q1,me1,'LineWidth',2,'Color','k')
colorbar
shading interp
axis image;axis off

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.



```

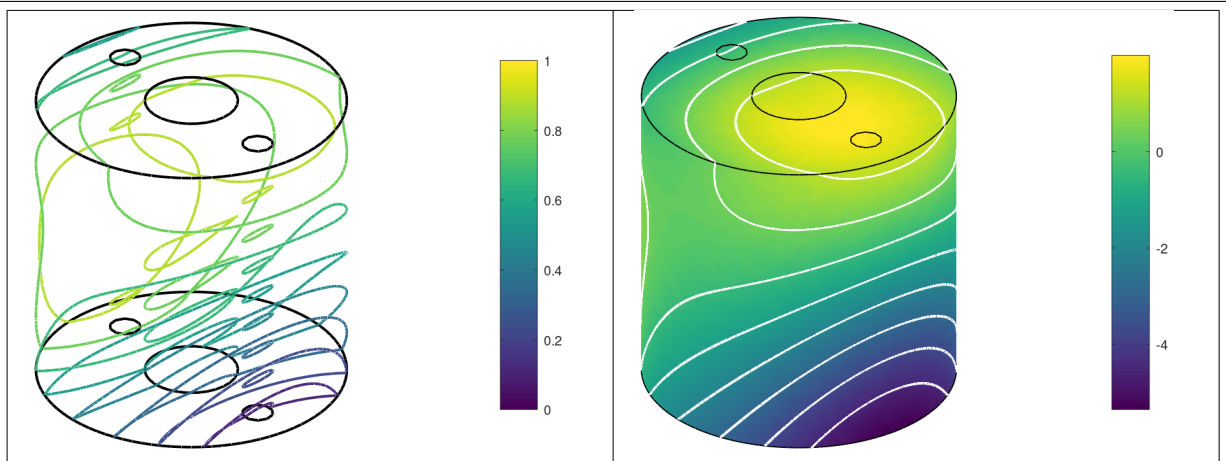
u=@(x,y,z) cos(x).*sin(y+z);
[q2,me2,toGlobal2]=fc_meshtools.simplicial.getMeshArrays('3Ds',2);
U2=fc_meshtools.eval(u,q2);
[q1,me1,toGlobal1]=fc_meshtools.simplicial.getMeshArrays('3Ds',1);
figure(1)
colormap('jet')
fc_graphics4mesh.plotmesh(q2,me2,'EdgeColor','LightGray','FaceColor','None')
hold on
fc_graphics4mesh.plotmesh(q1,me1,'Color','black')
fc_graphics4mesh.plotiso(q2,me2,U2,'isocolorbar',true,'format','%3f','LineWidth',2)
view(3);axis image;axis off

figure(2)
colormap('jet')
fc_graphics4mesh.plot(q2,me2,U2)
hold on
fc_graphics4mesh.plotmesh(q1,me1,'Color','black')
fc_graphics4mesh.plotiso(q2,me2,U2,'LineWidth',2,'Color','w')
colorbar

```

Listing 8: Using `fc_graphics4mesh.plotiso` function with a 3Ds mesh

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



```

u=@(x,y,z) -2+4*cos(x).*sin(y+z);
[q2,me2,toGlobal2]=fc_meshtools.simplicial.getMeshArrays('3D',2);
U2=fc_meshtools.eval(u,q2);
[q1,me1,toGlobal1]=fc_meshtools.simplicial.getMeshArrays('3D',1);
U1=fc_meshtools.eval(u,q1);
figure(1)
fc_graphics4mesh.plotmesh(q1,me1,'color','black','LineWidth',2)
hold on
fc_graphics4mesh.plotiso(q2,me2,U2,'LineWidth',2)
colorbar
shading interp
view(3);axis image;axis off
figure(2)
fc_graphics4mesh.plot(q2,me2,U2)
hold on
fc_graphics4mesh.plotiso(q2,me2,U2,'color','white','LineWidth',2)
fc_graphics4mesh.plotmesh(q1,me1,'color','black','LineWidth',1)
shading interp
view(3);axis image;axis off

```

Listing 9: Using `fc_graphics4mesh.plotiso` function with a 3D mesh

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 $P = \text{fc_tools.graphics.PlaneCoefs}(Q,V)$ 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.

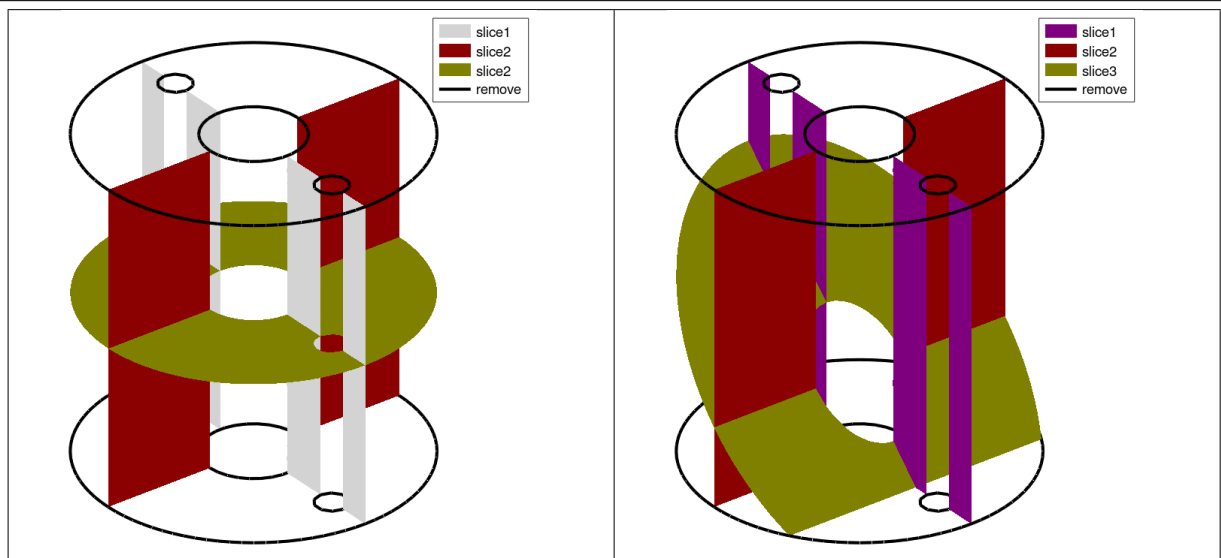
`slicemesh(q,me,P,Name,Value,...)` specifies function options using one or more `Name,Value` pair arguments.

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.



```
[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)
view(3);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)
view(3);axis off; axis image
```

Listing 10: Using `fc_graphics4mesh.slicemesh` function with a 3D mesh

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

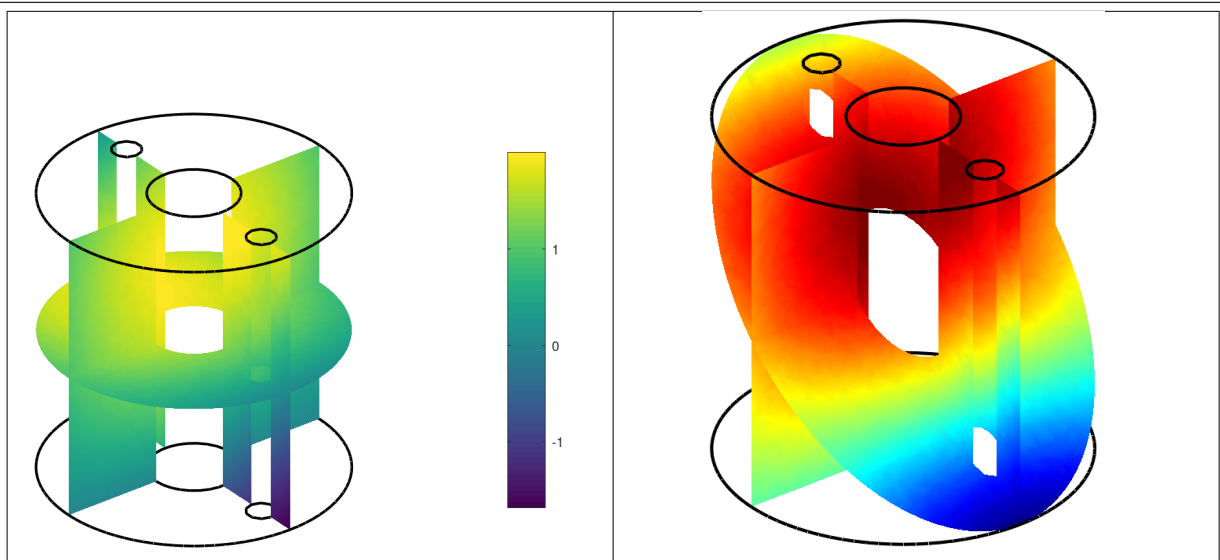
```
fc_graphics4mesh.slice(q,me,u,P)
fc_graphics4mesh.slice(q,me,u,P,Name,Value,...)
```

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 $P = \text{fc_tools.graphics.PlaneCoefs}(Q,V)$ 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 : the following code is part of the `fc_graphics4mesh.demos.slice3D` 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 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)
view(3);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)
```

Listing 11: Using `fc_graphics4mesh.slice` function with a 3D mesh

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 $P = \text{fc_tools.graphics.PlaneCoefs}(Q,V)$ 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

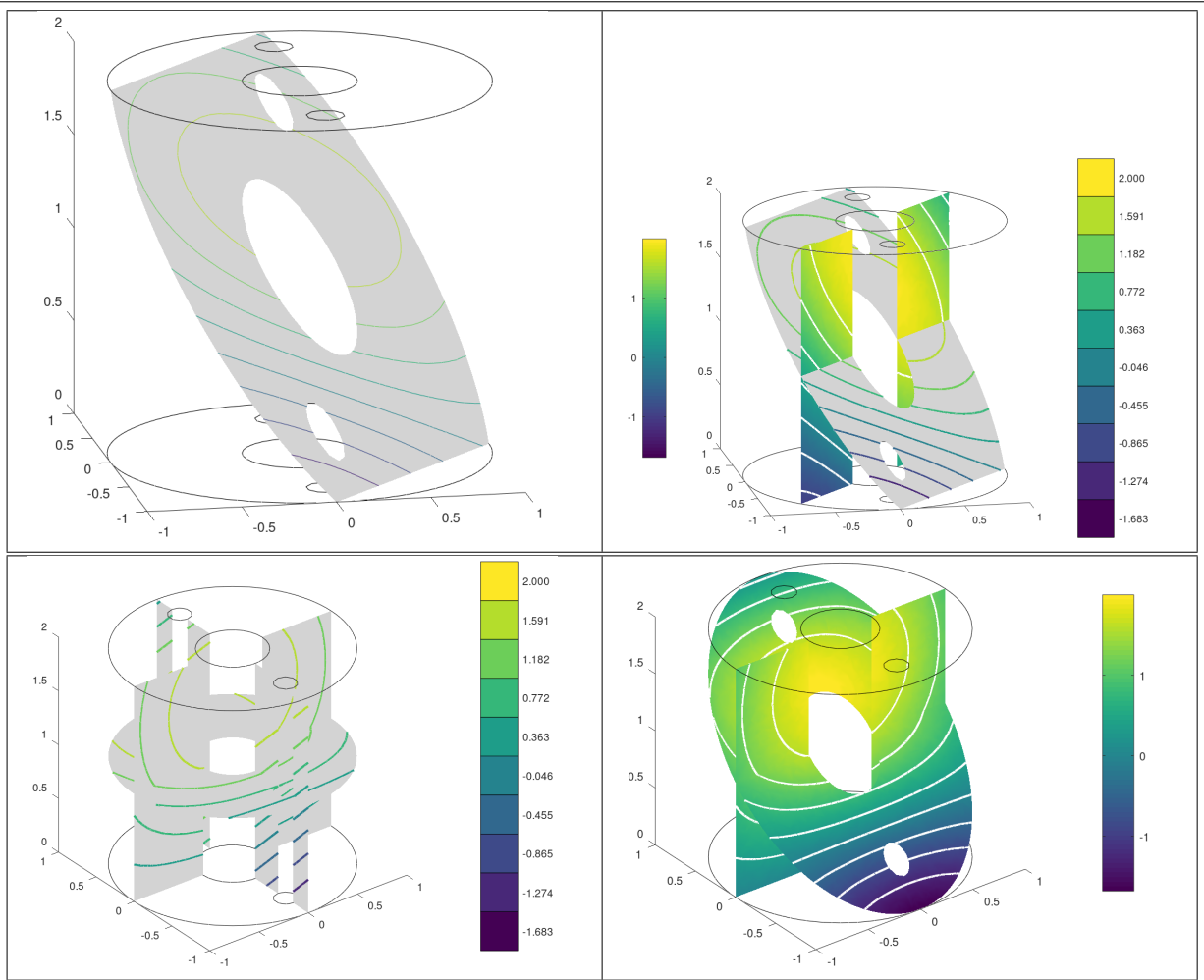
- `'niso'` : to specify the number of isolines (default : 10)
- `'isorange'` : to specify the list of isovalues (default : empty)
- `'color'` : to specify one color for all isolines (default : empty)
- `'isocolorbar'` : if true display a colorbar. Default is false.
- `'format'` : to specify the format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

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');
view(3);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)])
view(3);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 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

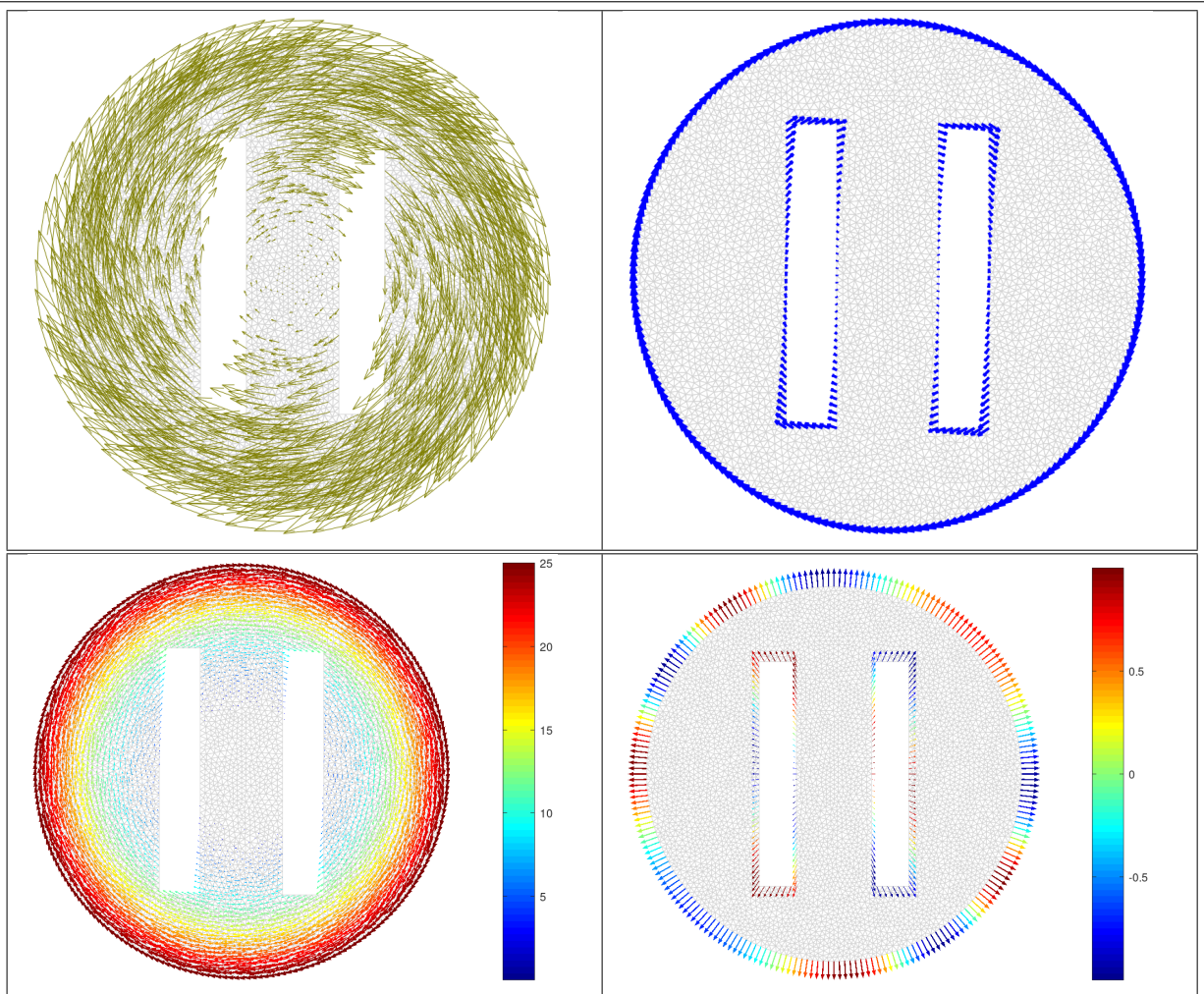
- `'freq'` : quiver frequency, (default : 1)
- `'scale'` : quiver scale, (default is `fc_meshtools.getCharacteristicLength(q)/20`)
- `'color'` : set one color for all quivers (default: default color of the **quiver** or `quiver3` functions). Cannot be used with `'colordata'` option.
- `'colordata'` : 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



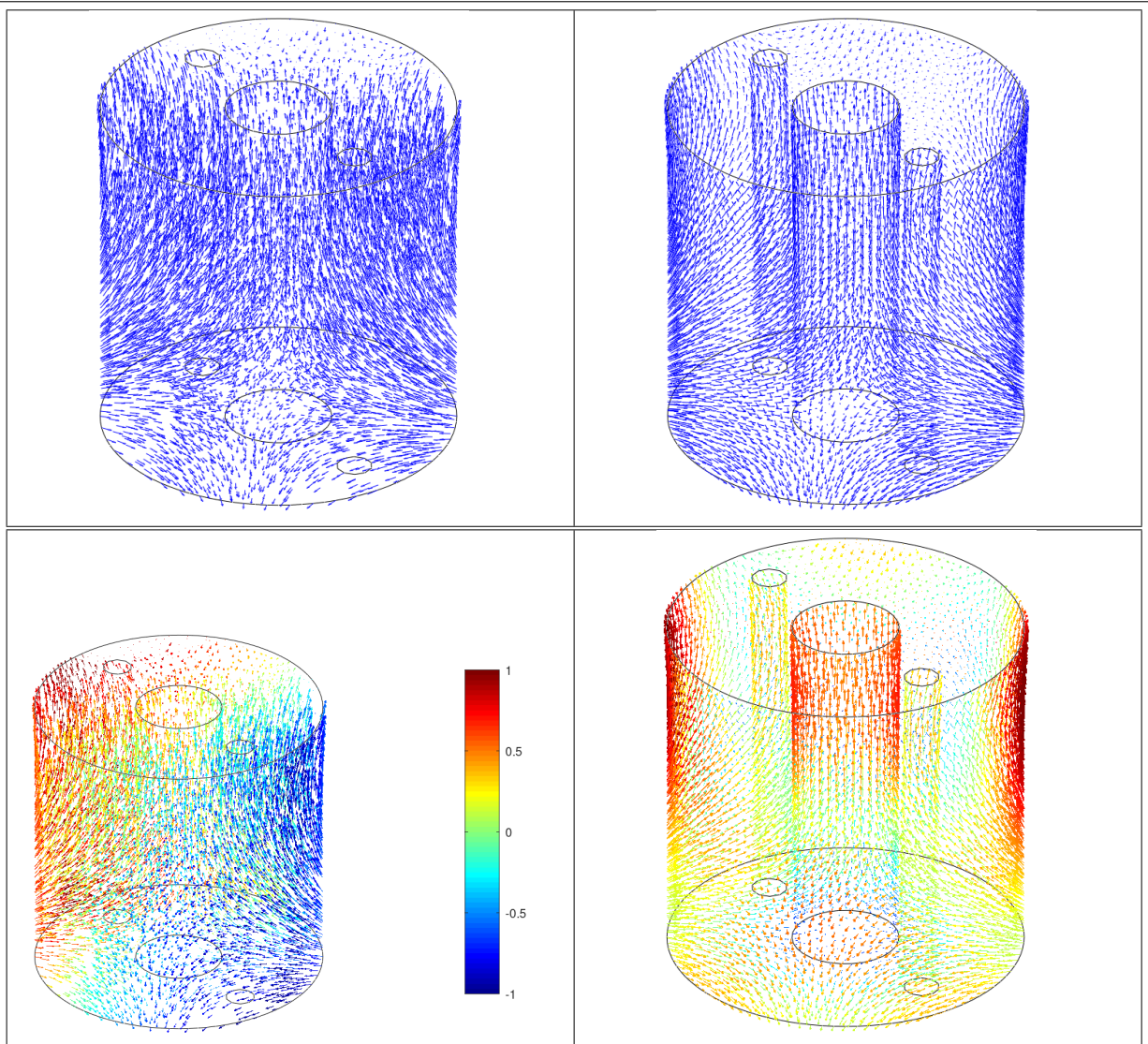
```

v=@(x,y) y.*sqrt(x.^2+y.^2),@(x,y) -x.*sqrt(x.^2+y.^2);
vv=@(x,y) x/5,@(x,y) y/5;
[q2,me2,toGlobal2]=fc_meshtools.simplicial.getMesh2D(2); % Select small mesh
V2=fc_meshtools.eval(v,q2);
u2=sqrt(V2{1}.^2+V2{2}.^2);
[q1,me1,toGlobal1]=fc_meshtools.simplicial.getMesh2D(1);
V1=fc_meshtools.eval(v,q1);
VV1=fc_meshtools.eval(vv,q1);
u1=sqrt(V1{1}.^2+V1{2}.^2);
figure(1)
fc_graphics4mesh.plotmesh(q2,me2,'color','LightGray')
hold on
fc_graphics4mesh.plotquiver(q2,me2,V2,'color','Olive','freq',4,'scale',0.1)
axis image;axis off
figure(2)
fc_graphics4mesh.plotmesh(q2,me2,'color','LightGray')
hold on
fc_graphics4mesh.plotquiver(q1,me1,V1,'LineWidth',2)
axis image;axis off
figure(3)
colormap('jet')
fc_graphics4mesh.plotmesh(q2,me2,'color','LightGray')
hold on
fc_graphics4mesh.plotquiver(q2,me2,V2,'colordata',u2)
axis image;axis off
colorbar
figure(4)
colormap('jet')
fc_graphics4mesh.plotmesh(q2,me2,'color','LightGray')
hold on
fc_graphics4mesh.plotquiver(q1,me1,VV1,'LineWidth',2,'colordata',@(x,y) sin(x+y))
axis image;axis off
colorbar

```

Listing 13: Plot quiver on 2D mesh

3D example



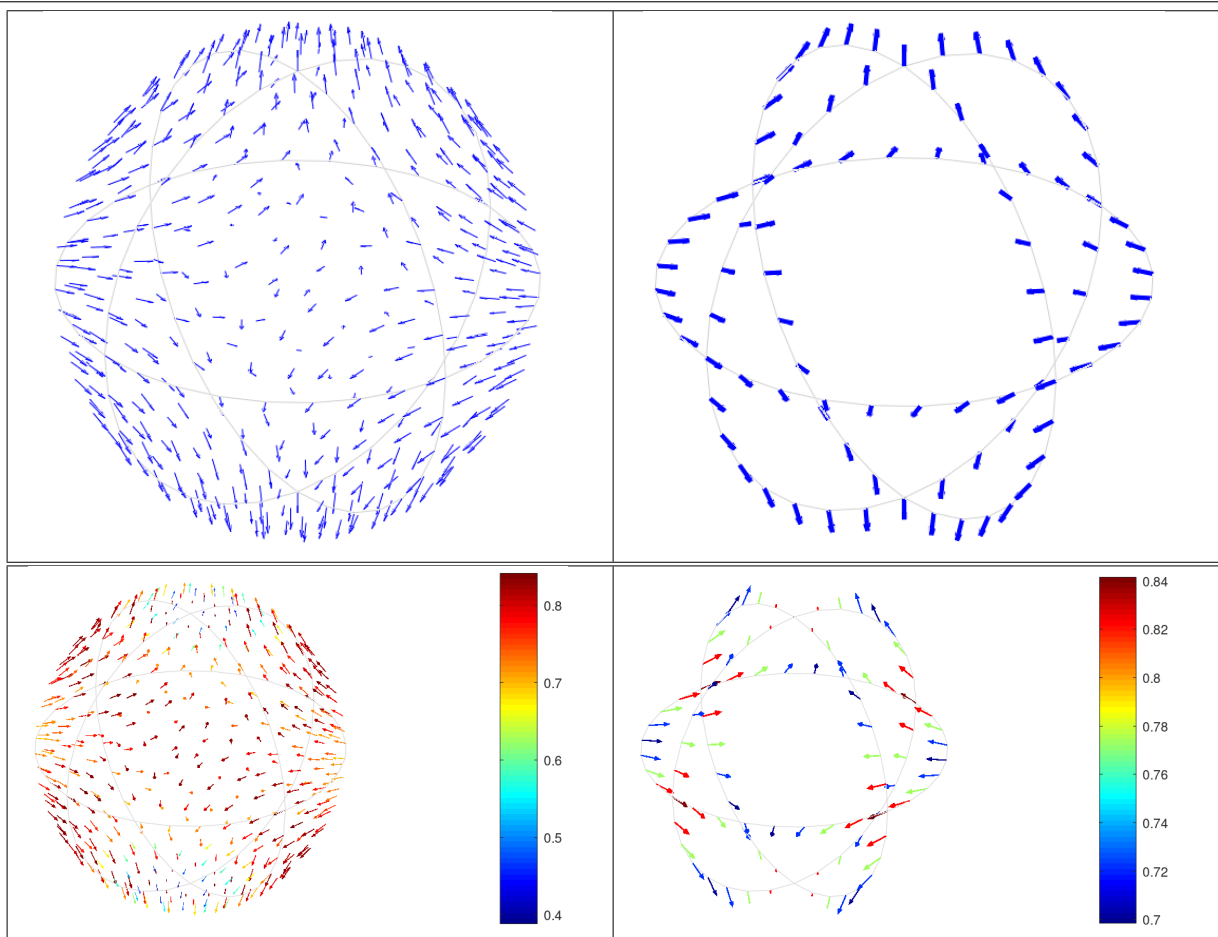
```

v=@(x,y,z) cos(x+z).*sin(y),@(x,y,z) sin(x).*cos(z+y),@(x,y,z) sin(z).*cos(x+y));
[q3,me3,toGlobal3]=fc_meshtools.simplicial.getMesh3D(3);
V3=fc_meshtools.eval(v,q3);
fu3=@(x,y,z) y;
u3=fc_meshtools.eval(fu3,q3);
[q2,me2,toGlobal2]=fc_meshtools.simplicial.getMesh3D(2);
V2=fc_meshtools.eval(v,q2);
u2=sqrt(V2{1}.^2+V2{2}.^2+V2{3}.^2);
[q1,me1,toGlobal1]=fc_meshtools.simplicial.getMesh3D(1);
V1=fc_meshtools.eval(v,q1);
u1=sqrt(V1{1}.^2+V1{2}.^2+V1{3}.^2);
figure(1)
fc_graphics4mesh.plotmesh(q1,me1,'color','k')
hold on;axis image;axis off
fc_graphics4mesh.plotquiver(q3,me3,V3,'freq',2)
figure(2)
fc_graphics4mesh.plotmesh(q1,me1,'color','k')
hold on;axis image;axis off
fc_graphics4mesh.plotquiver(q2,me2,V2)
figure(3)
colormap('jet')
fc_graphics4mesh.plotmesh(q1,me1,'color','k')
hold on;axis image;axis off
fc_graphics4mesh.plotquiver(q3,me3,V3,'colordata',u3,'freq',2)
colorbar
figure(4)
colormap('jet')
fc_graphics4mesh.plotmesh(q1,me1,'color','k')
hold on;axis image;axis off
fc_graphics4mesh.plotquiver(q2,me2,V2,'colordata',u2)%,'scale',1)

```

Listing 14: Plot quivers on 3D mesh

3D surface example



```

v=@(x,y,z) cos(x+z).*sin(y),@(x,y,z) sin(x).*cos(z+y),@(x,y,z) sin(z).*cos(x+y));
[q2,me2,toGlobal2]=fc_meshtools.simplicial.getMesh3Ds(2);
V2=fc_meshtools.eval(v,q2);
u2=sqrt(V2{1}.^2+V2{2}.^2+V2{3}.^2);
[q1,me1,toGlobal1]=fc_meshtools.simplicial.getMesh3Ds(1);
V1=fc_meshtools.eval(v,q1);
u1=sqrt(V1{1}.^2+V1{2}.^2+V1{3}.^2);
figure(1)
fc_graphics4mesh.plotmesh(q1,me1,'color','LightGray')
hold on
fc_graphics4mesh.plotquiver(q2,me2,V2)
axis image;axis off
figure(2)
fc_graphics4mesh.plotmesh(q1,me1,'color','LightGray')
hold on
fc_graphics4mesh.plotquiver(q1,me1,V1,'LineWidth',2)
axis image;axis off
figure(3)
colormap('jet')
fc_graphics4mesh.plotmesh(q1,me1,'color','LightGray')
hold on;axis image;axis off
fc_graphics4mesh.plotquiver(q2,me2,V2,'colordata',u2)
colorbar
figure(4)
colormap('jet')
fc_graphics4mesh.plotmesh(q1,me1,'color','LightGray')
hold on;axis image;axis off
fc_graphics4mesh.plotquiver(q1,me1,V1,'LineWidth',2,'colordata',u1,'scale',0.2)
colorbar

```

Listing 15: 3D surface mesh : plotquiver function

11 fc_graphics4mesh.scatter function

The function `fc_graphics4mesh.scatter` displays data on nodes array `q`

Syntax

```

fc_graphics4mesh.scatter(q,u)
fc_graphics4mesh.scatter(q,u,Name,Value,...)

```

Description

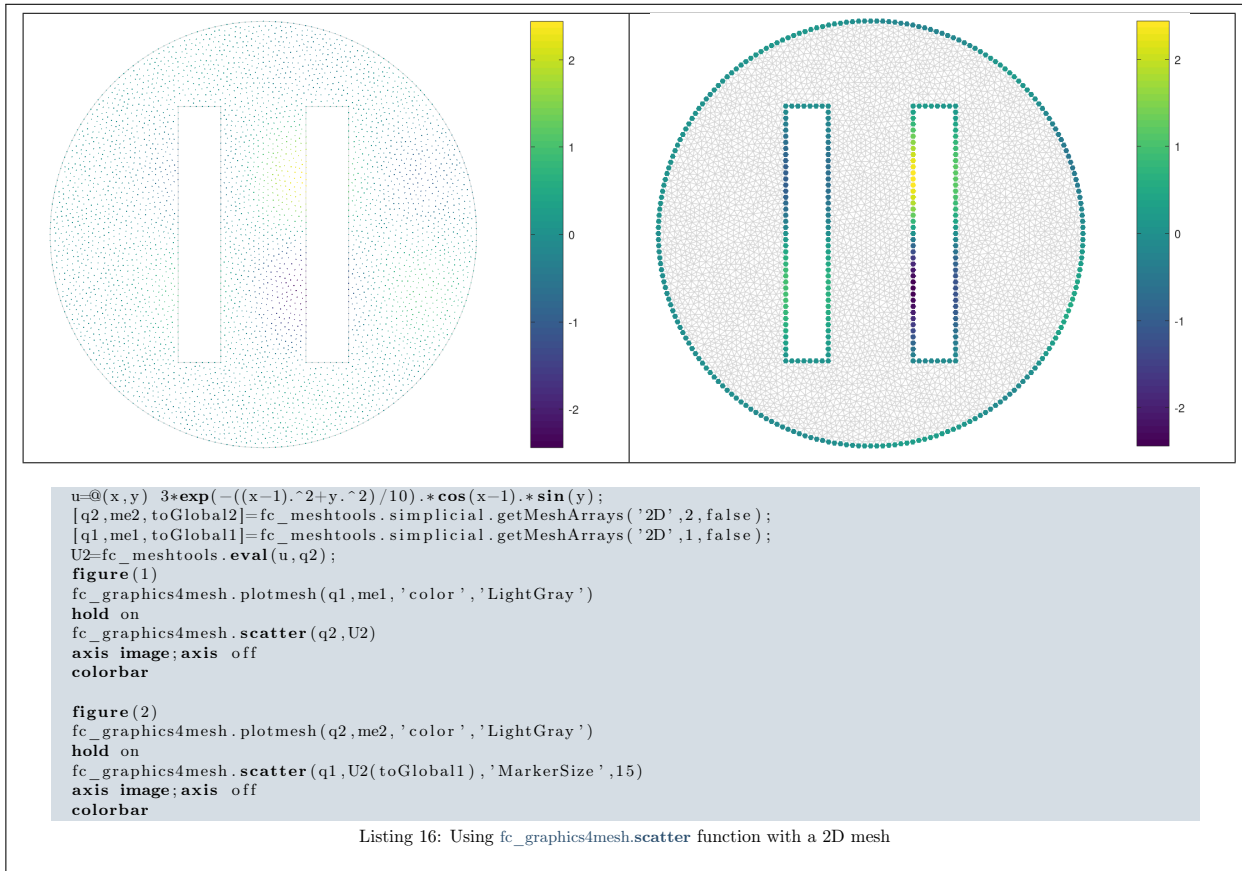
`scatter(q,u)` displays data `u` on nodes array `q` as colored points. The data `u` can be an handle function or an array.

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

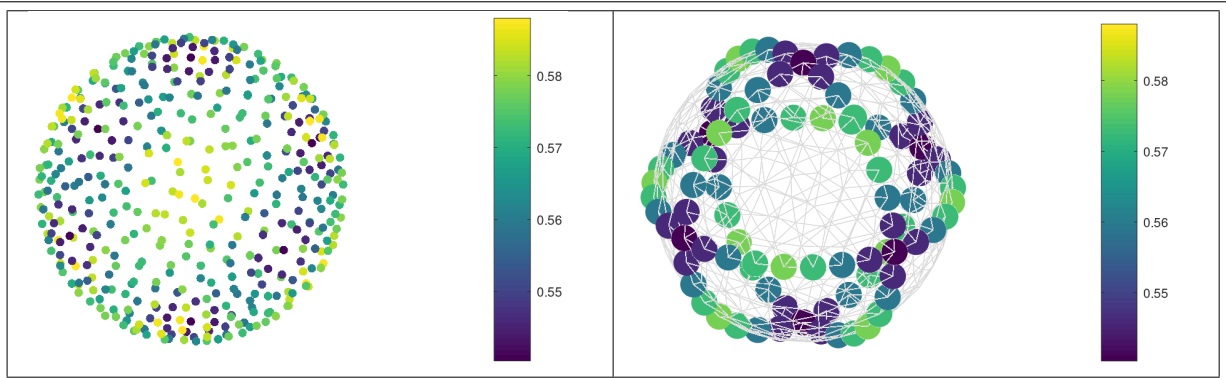
- `'MarkerSize'` : size of the marker. Default is 1.
- `'ForcePatch'` : if `true`, uses `patch` function, otherwise uses `scatter` function in dimension 2 or `scatter3` function in dimension 3. Default is `true`.

The options of second level are those of the function used (see `'ForcePatch'` option).

2D example : the following code is part of the `fc_graphics4mesh.demos.scatter2D` function.



3Ds example : the following code is part of the `fc_graphics4mesh.demos.scatter3Ds` function.



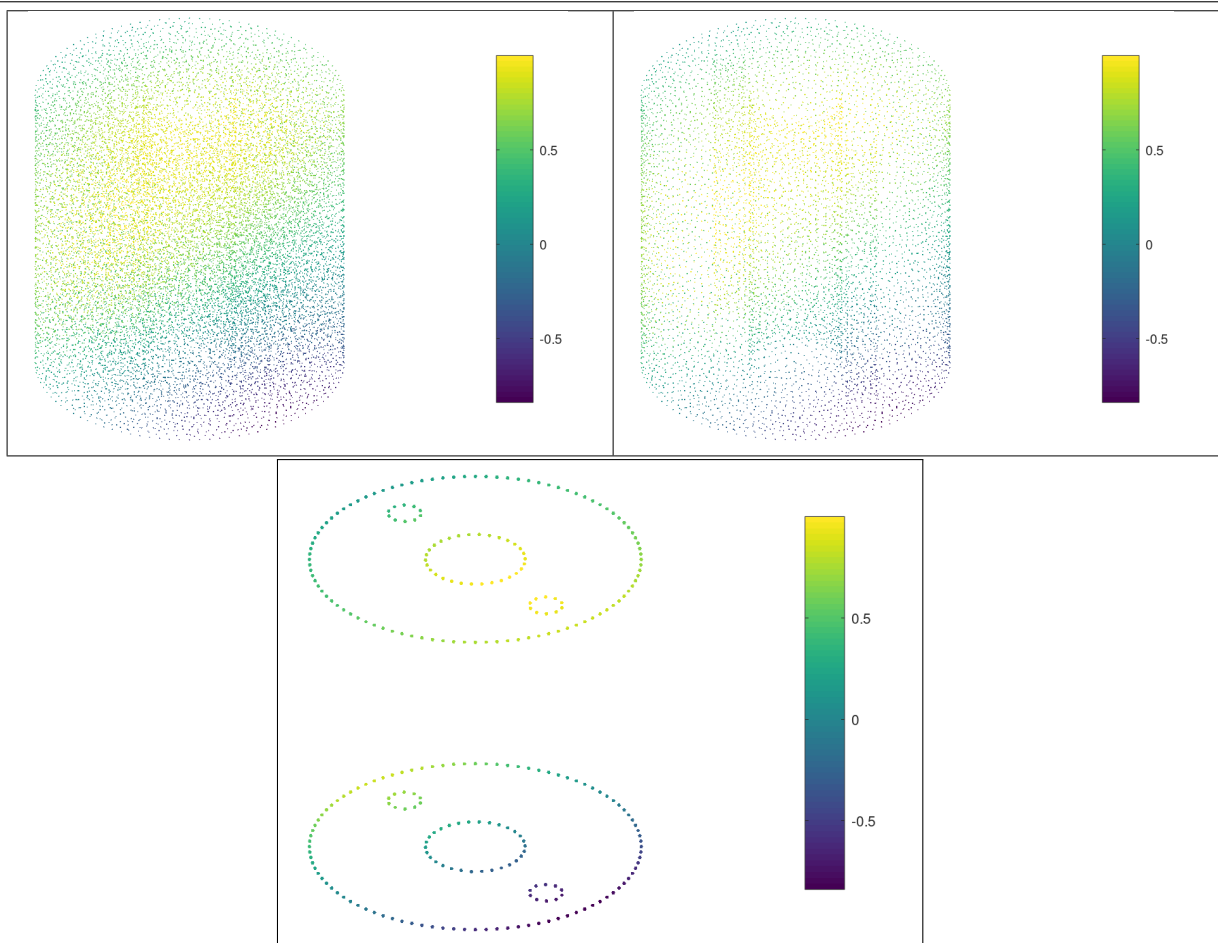
```

u=@(x,y,z) cos(x).*cos(y).*cos(z);
small=fc_tools.comp.isOctave();
[q2,me2,toGlobal2]=fc_meshtools.simplicial.getMesh3Ds(2,false);
[q1,me1,toGlobal1]=fc_meshtools.simplicial.getMesh3Ds(1,false);
U2=fc_meshtools.eval(u,q2);
U1=fc_meshtools.eval(u,q1);
figure(1)
fc_graphics4mesh.scatter(q2,U2,'Marker','.', 'MarkerSize',20)
view(3);axis image;axis off
colorbar
figure(2)
fc_graphics4mesh.plotmesh(q2,me2,'EdgeColor','LightGray','FaceColor','none')
hold on
fc_graphics4mesh.scatter(q1,U1,'Marker','o','MarkerFaceColor','flat','MarkerSize',20)
view(3);axis image;axis off
caxis([min(U2),max(U2)])
colorbar

```

Listing 17: Using `fc_graphics4mesh.scatter` function with a 3Ds mesh

3D example : the following code is part of the `fc_graphics4mesh.demos.scatter3D` function.



```

u=@(x,y,z) cos(x).*sin(y+z);
small=~fc_tools.comp.isOctave();
[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.scatter(q3,U3,'MarkerEdgeAlpha',0.3)
view(3);axis image;axis off
colorbar

figure(2)
fc_graphics4mesh.scatter(q2,U3(toGlobal2))
hold on
fc_graphics4mesh.plotmesh(q1,me1,'color','black','LineWidth',2)
view(3);axis image;axis off
caxis([min(U3),max(U3)])
colorbar

figure(3)
fc_graphics4mesh.plotmesh(q2,me2,'EdgeColor','LightGray','FaceColor','None','EdgeAlpha',0.2)
hold on
fc_graphics4mesh.scatter(q1,U3(toGlobal1),'MarkerFaceColor','flat','MarkerSize',8)
view(3);axis image;axis off
caxis([min(U3),max(U3)])
colorbar

```

Listing 18: Using `fc_graphics4mesh.scatter` function with a 3D mesh

12 `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.

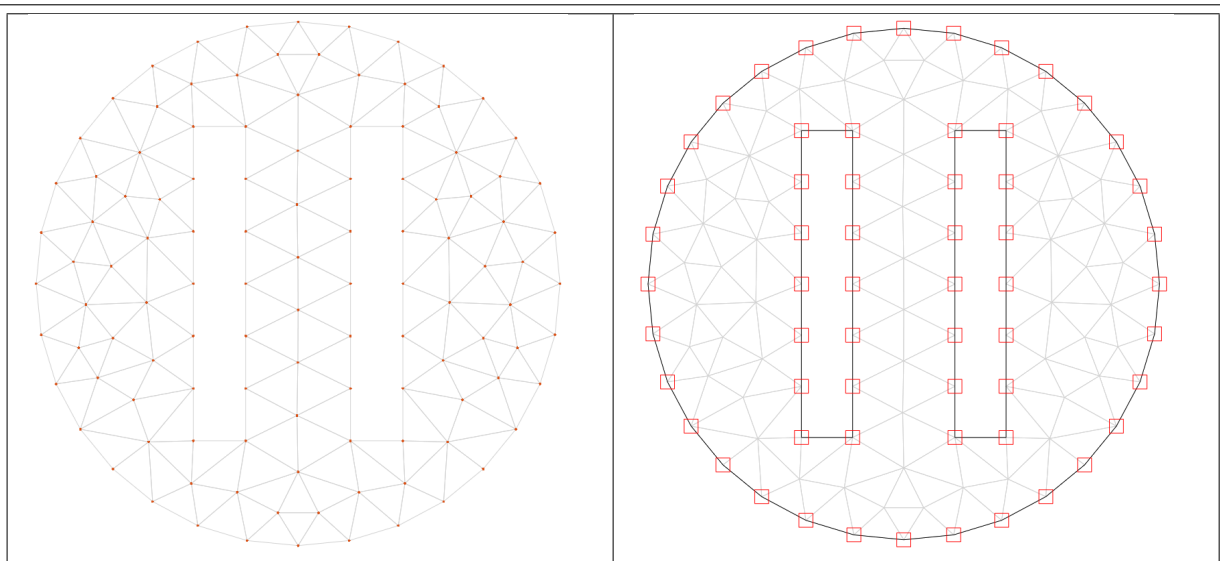
`plotnodes(q,Name,Value, ...)` specifies options using one or more `Name,Value` pair arguments. Options of first level are

- `'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.

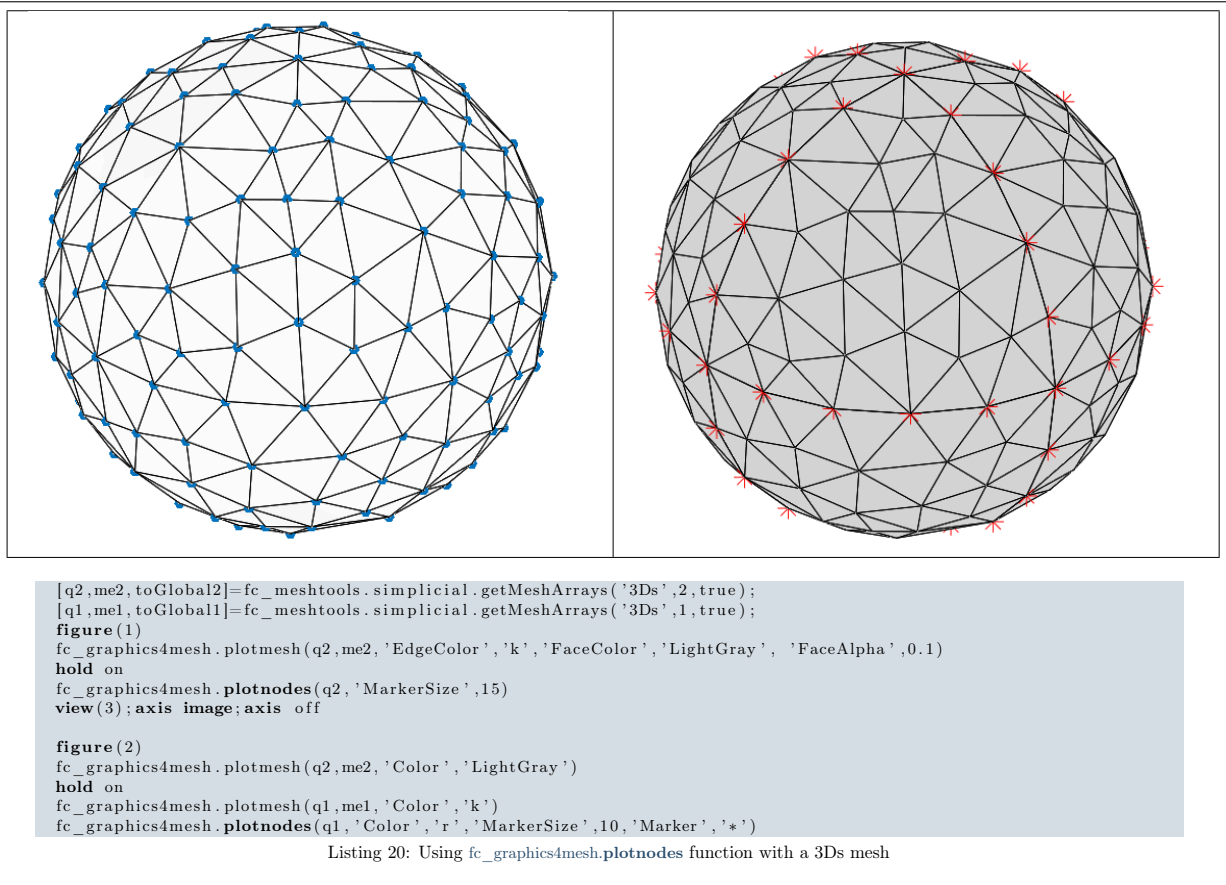


```
[q2,me2,toGlobal2]=fc_meshtools.simplicial.getMeshArrays('2D',2,true);
[q1,me1,toGlobal1]=fc_meshtools.simplicial.getMeshArrays('2D',1,true);
figure(1)
fc_graphics4mesh.plotmesh(q2,me2,'Color','LightGray')
hold on
fc_graphics4mesh.plotnodes(q2)
axis image;axis off

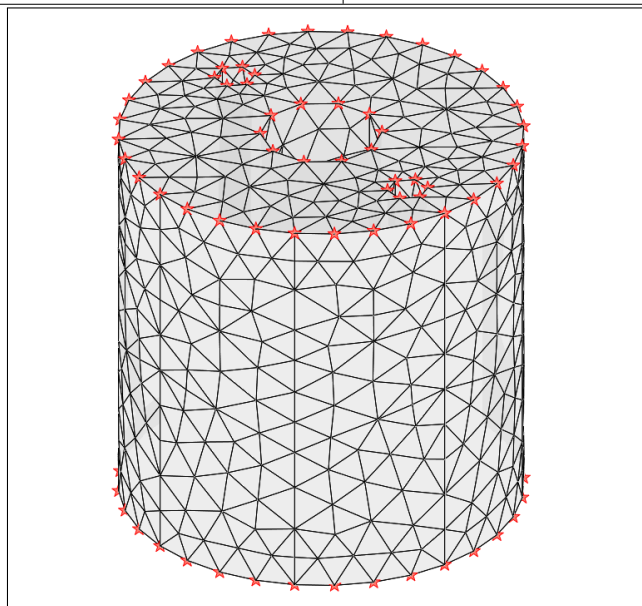
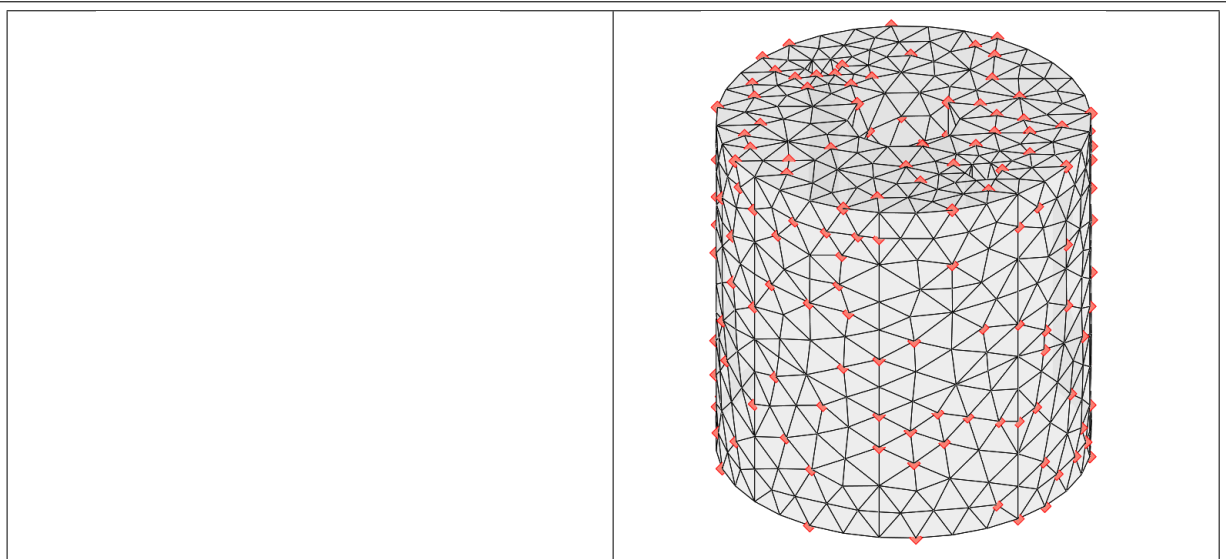
figure(2)
fc_graphics4mesh.plotmesh(q2,me2,'Color','LightGray')
hold on
fc_graphics4mesh.plotmesh(q1,me1,'Color','k')
fc_graphics4mesh.plotnodes(q1,'Color','r','MarkerSize',10,'Marker','s')
```

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

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.



```
[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
fc_graphics4mesh.plotnodes(q3,'Color','r','MarkerSize',10)
view(3);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.plotnodes(q2,'idx',idx,'MarkerSize',8,'Marker','d','MarkerEdgeColor',...
    'red','MarkerFaceColor','Salmon')
view(3);axis image;axis off

figure(3)
fc_graphics4mesh.plotmesh(q2,me2,'FaceColor','LightGray','FaceAlpha',0.4)
hold on
fc_graphics4mesh.plotnodes(q1,'MarkerSize',8,'Marker','p','MarkerEdgeColor','red','MarkerFaceColor','Salmon')
```

Listing 21: Using `fc_graphics4mesh.plotnodes` function with a 3D mesh

13 `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

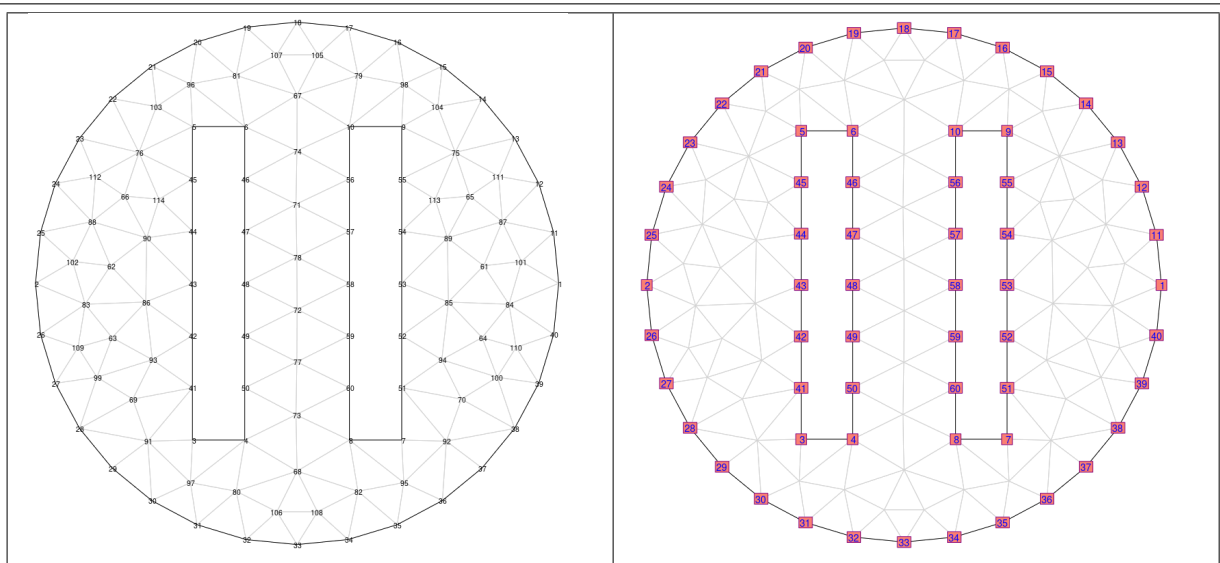
`plotnodesidx(q)` displays all the numbers/indices of the nodes array `q`

`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.

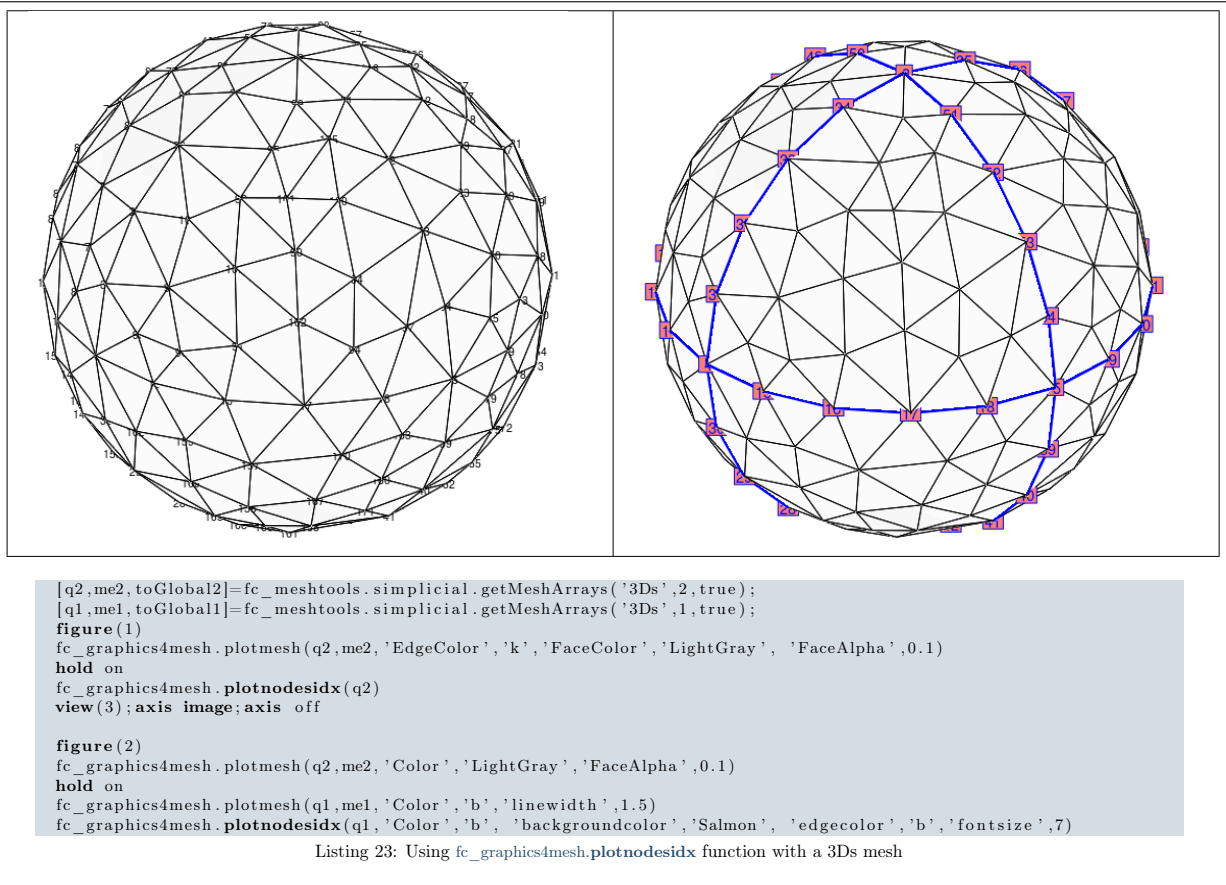


```
[q2,me2,toGlobal2]=fc_meshtools.simplicial.getMeshArrays('2D',2,true);
[q1,me1,toGlobal1]=fc_meshtools.simplicial.getMeshArrays('2D',1,true);
figure(1)
fc_graphics4mesh.plotmesh(q2,me2,'Color','LightGray')
hold on
fc_graphics4mesh.plotmesh(q1,me1,'Color','k')
fc_graphics4mesh.plotnodesidx(q2)
axis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

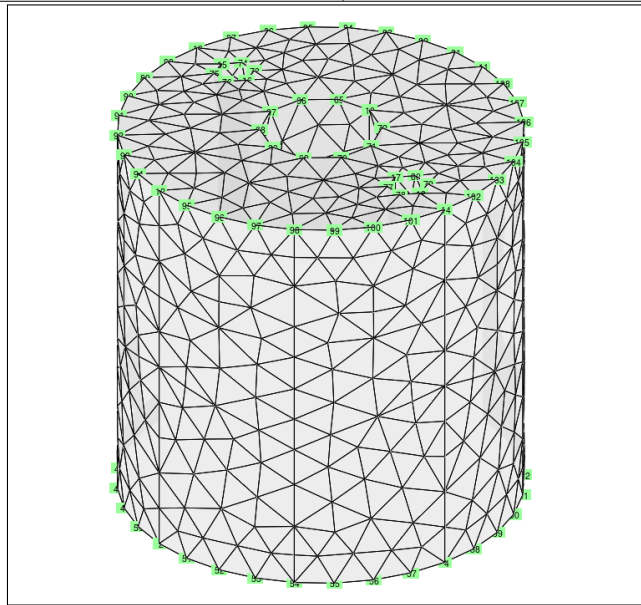
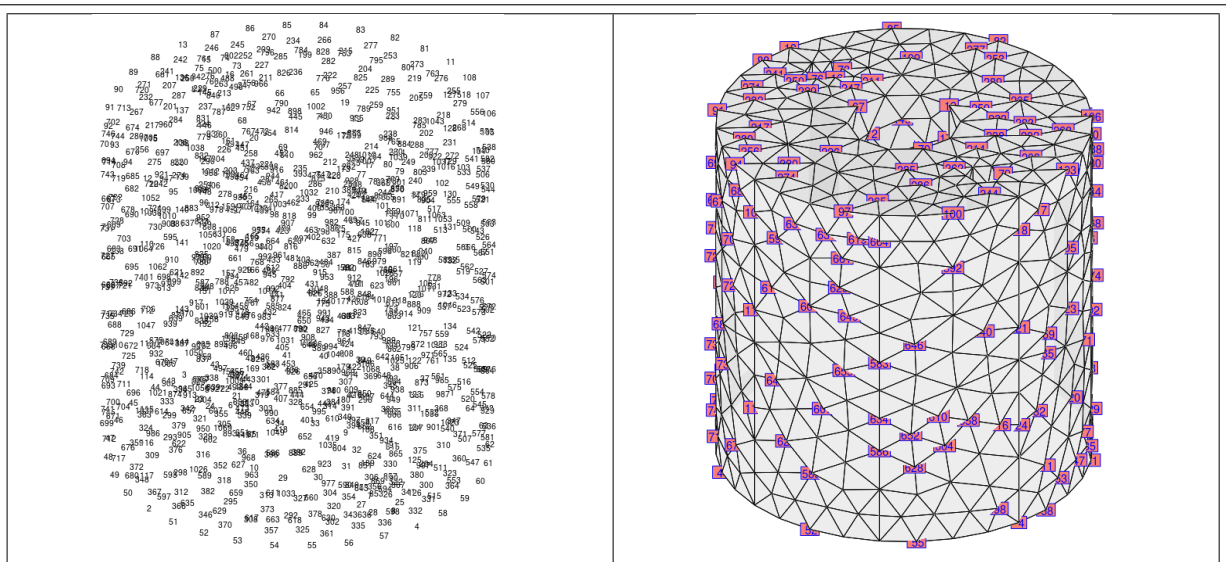
figure(2)
fc_graphics4mesh.plotmesh(q2,me2,'Color','LightGray')
hold on
fc_graphics4mesh.plotmesh(q1,me1,'Color','k')
fc_graphics4mesh.plotnodesidx(q1,'Color','b','backgroundcolor','Salmon','edgecolor','purple','fontSize',7)
```

Listing 22: 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.



```
[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
fc_graphics4mesh.plotnodesidx(q3)
view(3);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.plotnodesidx(q2,'idx',idx,'Color','b','backgroundcolor','Salmon',...
'edgecolor','b','fontsize',7)
view(3);axis image;axis off

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

Listing 24: Using `fc_graphics4mesh.plotnodesidx` function with a 3D mesh

14 `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`.

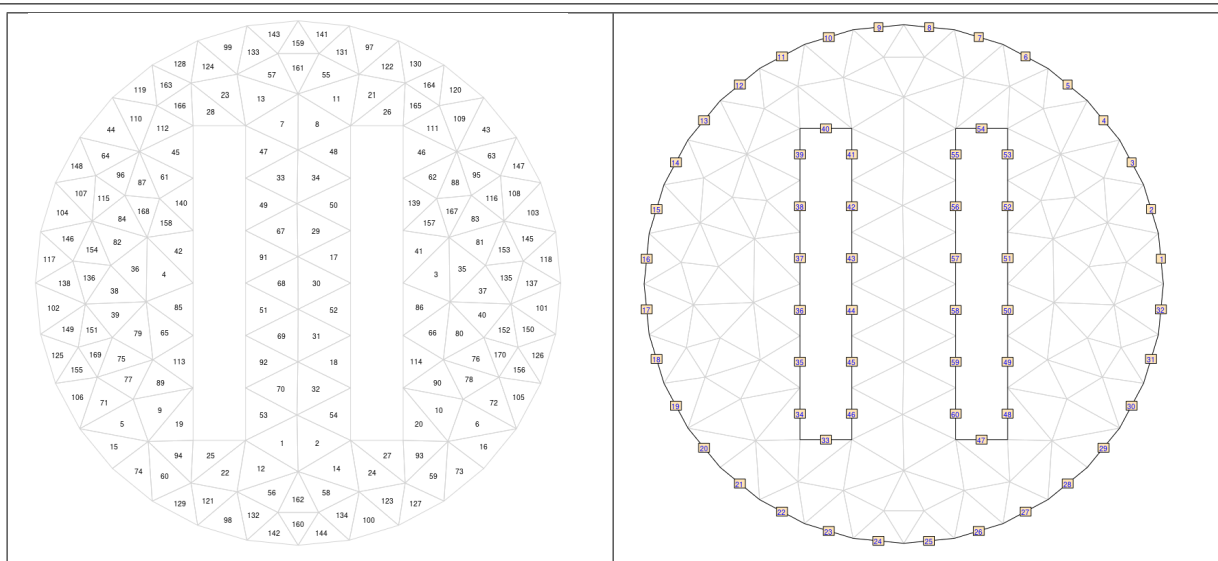
`plotelementsidx(q,me,Name,Value,...)` specifies function options using one or more `Name,Value` pair arguments.

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.

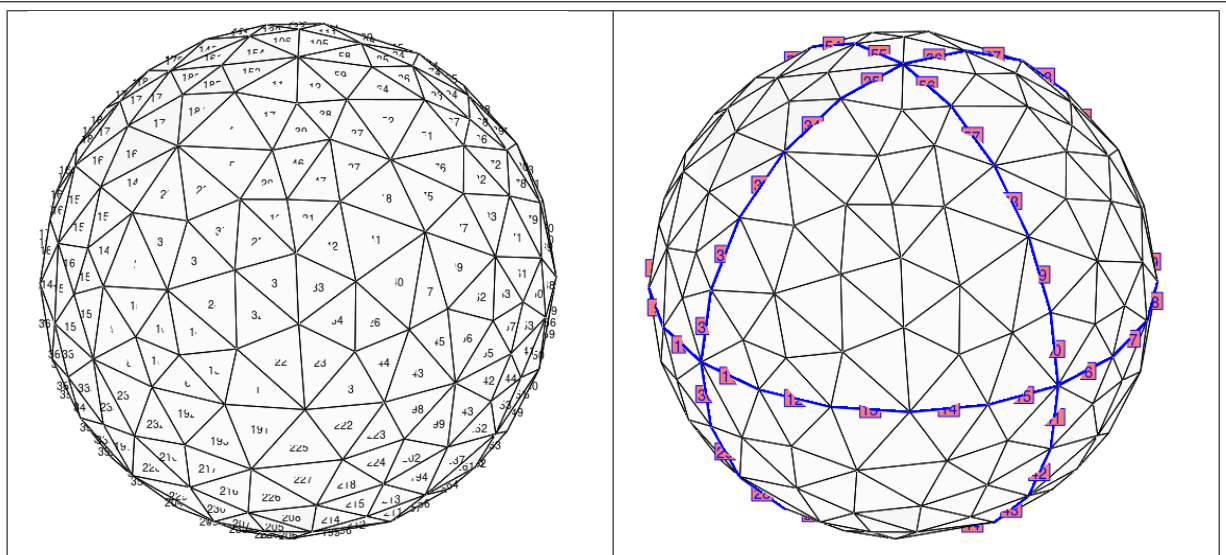


```
[q2,me2,toGlobal2]=fc_meshtools.simplicial.getMeshArrays('2D',2,true);
[q1,me1,toGlobal1]=fc_meshtools.simplicial.getMeshArrays('2D',1,true);
figure(1)
fc_graphics4mesh.plotmesh(q2,me2,'Color','LightGray')
hold on
fc_graphics4mesh.plotelementsidx(q2,me2)
axis image;axis off

figure(2)
fc_graphics4mesh.plotmesh(q2,me2,'Color','LightGray')
hold on
fc_graphics4mesh.plotmesh(q1,me1,'Color','k')
fc_graphics4mesh.plotelementsidx(q1,me1,'Color','b','BackgroundColor','Moccasin','FontSize',5,'edgecolor','k')
```

Listing 25: Using `fc_graphics4mesh.plotelementsidx` function with a 2D mesh

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

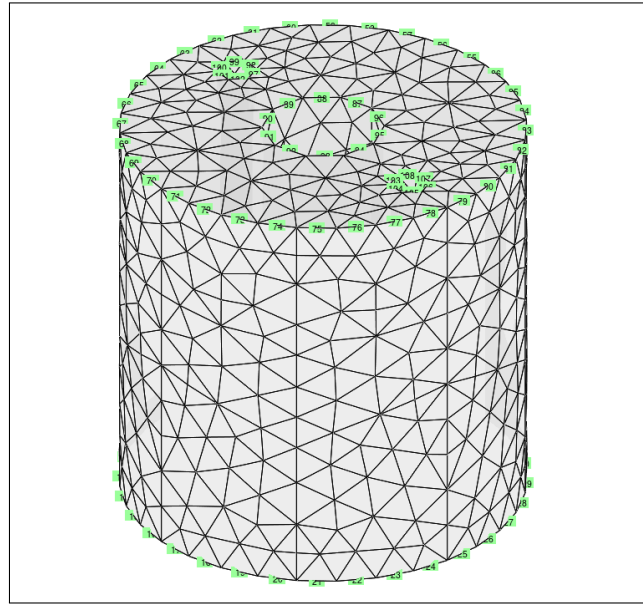
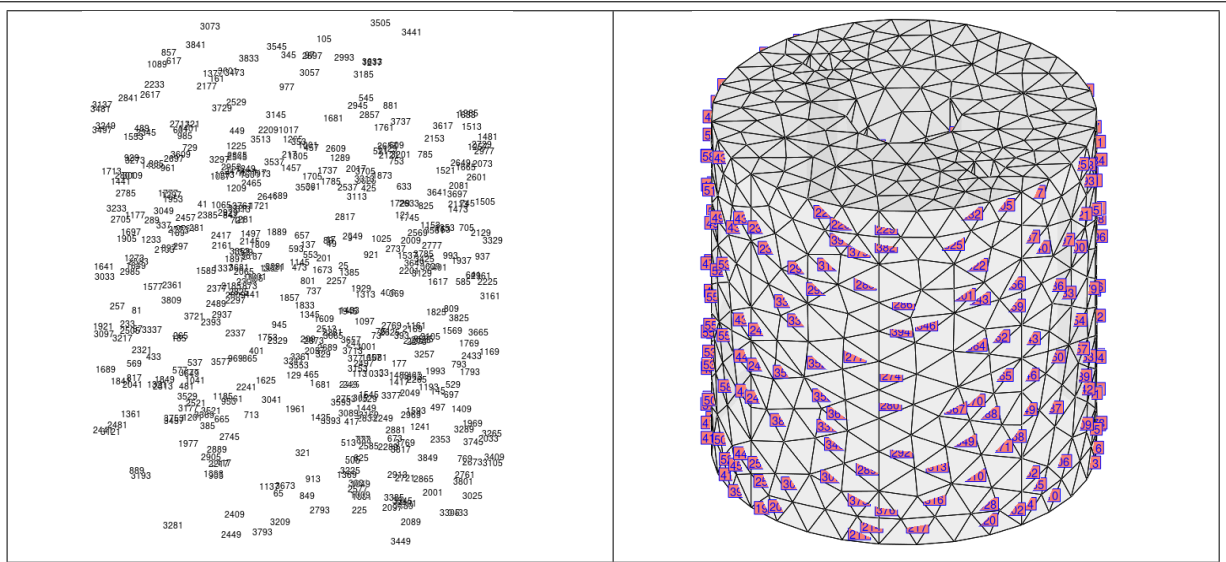


```
[q2,me2,toGlobal2]=fc_meshtools.simplicial.getMeshArrays('3Ds',2,true);
[q1,me1,toGlobal1]=fc_meshtools.simplicial.getMeshArrays('3Ds',1,true);
figure(1)
fc_graphics4mesh.plotmesh(q2,me2,'EdgeColor','k','FaceColor','LightGray','FaceAlpha',0.1)
hold on
fc_graphics4mesh.plotelementsidx(q2,me2)
view(3);axis image;axis off

figure(2)
fc_graphics4mesh.plotmesh(q2,me2,'Color','LightGray','FaceAlpha',0.1)
hold on
fc_graphics4mesh.plotmesh(q1,me1,'Color','b','linewidth',1.5)
fc_graphics4mesh.plotelementsidx(q1,me1,'Color','b','backgroundcolor','Salmon','edgecolor','b','fontSize',7)
```

Listing 26: Using `fc_graphics4mesh.plotelementsidx` function with a 3Ds mesh

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)
view(3);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)
view(3);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 27: Using `fc_graphics4mesh.plotelementsidx` function with a 3D mesh

Informations for git maintainers of the graphics4mesh Octave package

git informations on the packages used to build this manual

```
-----
name : fc-graphics4mesh
tag : 0.1.6
commit : 184e91b337ead27cb05f945983446dc66ec02795
date : 2023-03-04
time : 08-38-29
status : 0
-----
name : fc-tools
tag : 0.0.35
commit : 1469d4cbac235c83155cf9195d9887e0a149177a
date : 2023-03-03
time : 13-50-59
status : 0
-----
name : fc-bench
tag : 0.1.3
commit : 0639dde1dee68427af4dab19f75b1c58dacf64a4
date : 2023-03-02
time : 09-35-44
status : 0
-----
name : fc-amat
tag : 0.1.3.b
commit : 36648a410d06d8af84f4950cd42681309d803dbf
date : 2023-03-02
time : 09-49-35
status : 0
-----
name : fc-meshtools
tag : 0.1.4.b
commit : 43f58ffcfa2ca359f2d948d156a8165274270376
date : 2023-03-02
time : 15-23-05
status : 0
-----
```

git informations on the L^AT_EX package used to build this manual

```
-----
name : fctools
tag :
commit : 72f48895e5cef4f2bf93bf6ef4f7e170382f03bf
date : 2023-03-02
time : 15:04:17
status : 1
-----
```

Using the remote configuration repository:

```
url      ssh://lagagit/MCS/Cuvelier/Matlab/fc-config
commit   ca906afbfb4703ef2dba4e43d9b4385267301e1
```