graphics4mesh Octave package, User's Guide*

version 0.1.0

François Cuvelier†

January 29, 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

*LaTeX manual, revision 0.1.0, compiled with Octave 5.1.0, and packages `fc-graphics4mesh`[0.1.0], `fc-tools`[0.0.29], `fc-bench`[0.1.1], `fc-amat`[0.1.1], `fc-meshtools`[0.1.2]

†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.

This package was tested on various OS with Octave releases:

	Octave					
Operating system	**4.2.0**	**4.2.1**	**4.2.2**	**4.4.0**	**4.4.1**	**5.1.0**
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				✓	✓	
MacOS Catalina 10.15.2				✓	✓	
Windows 10 (1909)	✓	✓	✓	✓	✓	✓

It is not compatible with Octave releases prior to 4.2.0. The Octave releases tested are those from

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

- **MacOS :** binaries from `http://octave-app.org/Download.html`;
- **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/0.1.0/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-0.1.0/configure_loc.m ...
     ...
3- Using packages :
   ->             fc-tools : 0.0.29
   ->             fc-bench : 0.1.1
   ->              fc-amat : 0.1.1
   ->         fc-meshtools : 0.1.2
 with     fc-graphics4mesh : 0.1.0
*** Using instructions
   To use the <fc-graphics4mesh> package:
   addpath('~/Octave/fc-graphics4mesh-full/fc_graphics4mesh-0.1.0')
   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-0.0.2')
>> 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 ...
    /home/cuvelier/Octave/fc-graphics4mesh-full/fc_tools-0.0.29/configure_loc.m ...
    ...
Try to use default parameters!
 Use fc_bench.configure to configure.
Write in ...
    /home/cuvelier/Octave/fc-graphics4mesh-full/fc_bench-0.1.1/configure_loc.m ...
    ...
Try to use default parameters!
 Use fc_amat.configure to configure.
Write in ...
    /home/cuvelier/Octave/fc-graphics4mesh-full/fc_amat-0.1.1/configure_loc.m ...
    ...
Try to use default parameters!
 Use fc_meshtools.configure to configure.
Write in ...
    /home/cuvelier/Octave/fc-graphics4mesh-full/fc_meshtools-0.1.2/configure_loc.m ...
    ...
Using fc_graphics4mesh[0.1.0] with fc_tools[0.0.29], fc_bench[0.1.1], ...
    fc_amat[0.1.1], fc_meshtools[0.1.2].
```

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

```
Using fc_graphics4mesh[0.1.0] with fc_tools[0.0.29], fc_bench[0.1.1], ...
    fc_amat[0.1.1], fc_meshtools[0.1.2].
```

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 $0 \leqslant d \leqslant dim$ 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).
- '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.

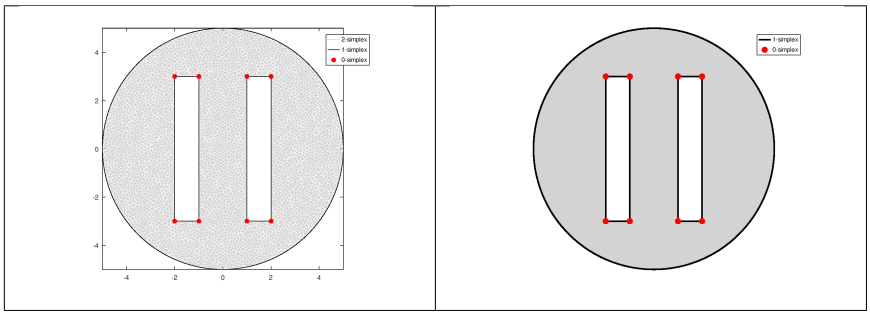
The options of second level depend on the type of elementaries mesh elements to represent.
One can use any option of the following functions according to the type of d-simplex to be represented.

- 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,

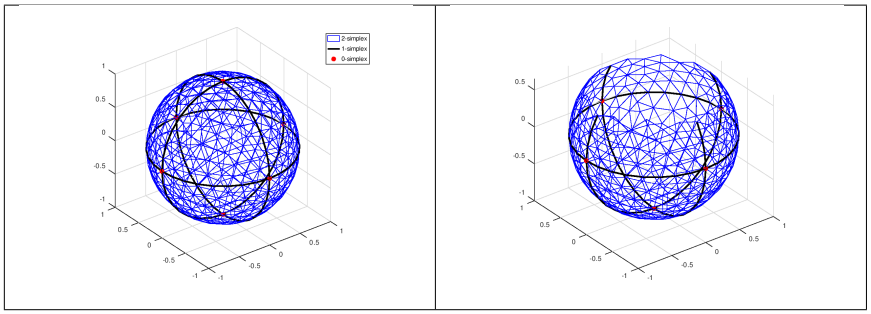
2D example : the following code is part of the `fc_graphics4mesh.demos.plotmesh2D` function.



```
[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
legend show

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

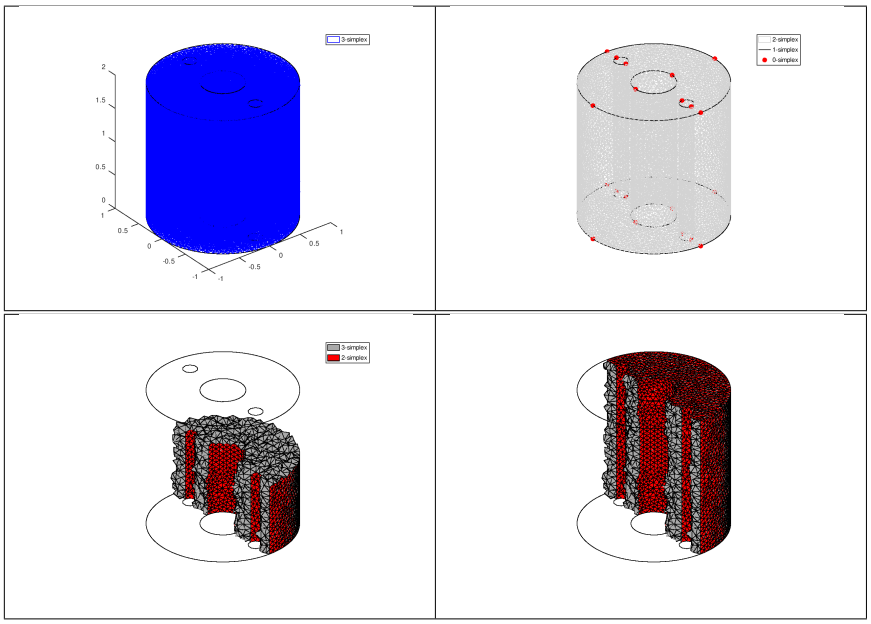
3Ds example : the following code is part of the `fc_graphics4mesh.demos.plotmesh3Ds` function.




```
[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)
view(3)
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')
axis image
legend show
P=fc_tools.graphics.PlaneCoefs([0 0 1/2], [0 0 1]);
figure(2)
fc_graphics4mesh.plotmesh(q2,me2,'EdgeColor','blue','FaceColor','None', ...
    'cutPlan',P)
hold on
fc_graphics4mesh.plotmesh(q1,me1,'color','black', 'Linewidth',2,'cutPlan',P)
fc_graphics4mesh.plotmesh(q0,me0,'color','red','cutPlan',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
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')
```

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

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

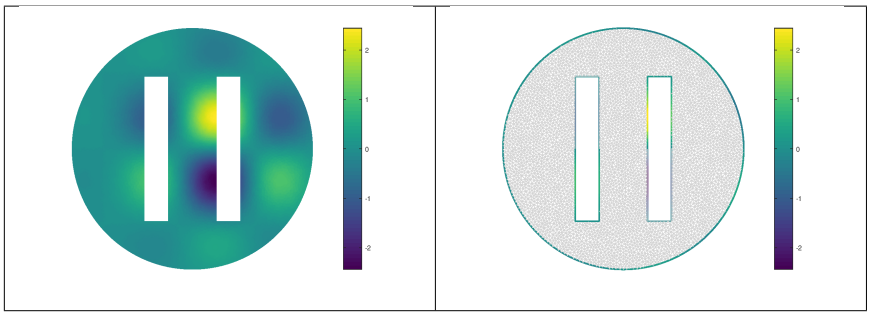
$$P(i,1)\,x + P(i,2)\,y + P(i,3)\,z + P(i,4) = 0.$$

  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 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 'plan' to true) function is used,
  - if $d == 1$, **patch** function is used,
- In dimension 1, **plot** function is used.

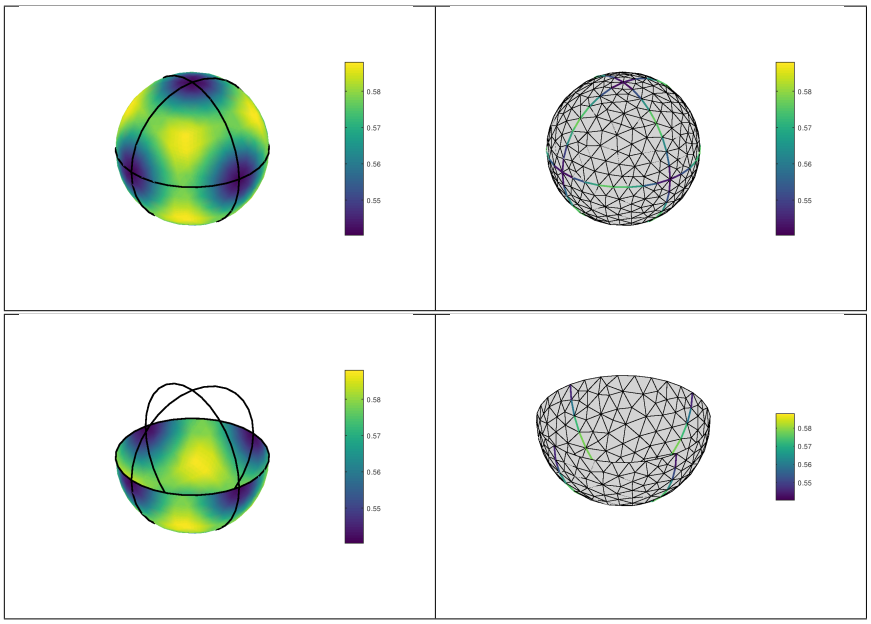
2D example : the following code is part of the `fc_graphics4mesh.demos.plot2D` function.



```
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,'plan',true)
shading interp
axis image;axis off
colorbar

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

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

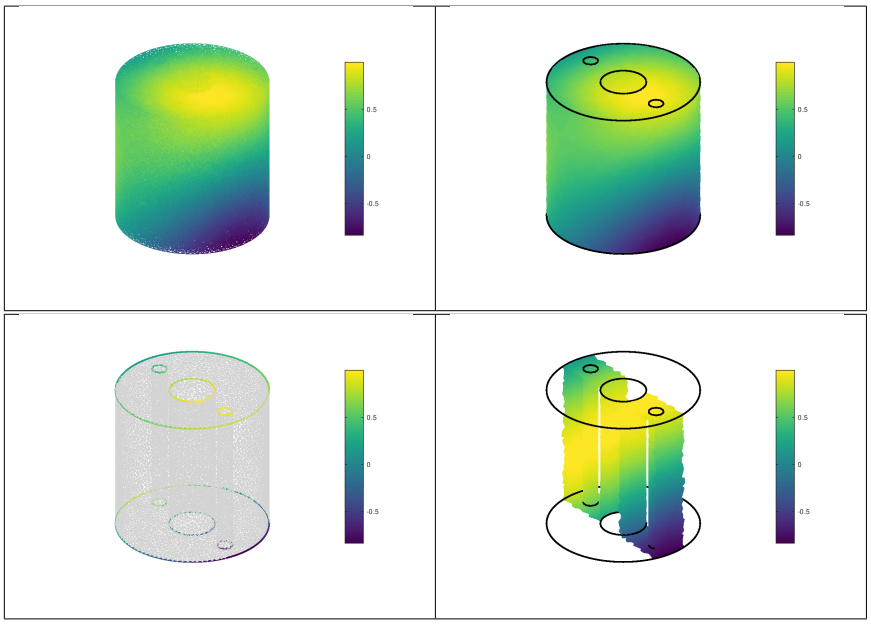
3Ds example : the following code is part of the `fc_graphics4mesh.demos.plot3Ds` function.



```
u=@(x,y,z) cos(x).*cos(y).*cos(z);
[q2,me2,toGlobal2]=fc_meshtools.simplicial.getMesh3Ds(2);
[q1,me1,toGlobal1]=fc_meshtools.simplicial.getMesh3Ds(1);
U2=fc_meshtools.eval(u,q2);
U1=fc_meshtools.eval(u,q1);
figure(1)
fc_graphics4mesh.plot(q2,me2,U2)
hold on
fc_graphics4mesh.plotmesh(q1,me1,'color','k','LineWidth',2)
shading interp
axis image;axis off
colorbar
figure(2)
fc_graphics4mesh.plotmesh(q2,me2,'color','LightGray')
hold on
fc_graphics4mesh.plot(q1,me1,U1,'LineWidth',2)
axis image;axis off
caxis([min(U2),max(U2)])
colorbar
P=fc_tools.graphics.PlaneCoefs([0 0 0], [0 0 1]);
figure(3)
fc_graphics4mesh.plot(q2,me2,U2,'cutPlan',P)
hold on
fc_graphics4mesh.plotmesh(q1,me1,'color','k','LineWidth',2)
shading interp
axis image;axis off
caxis([min(U2),max(U2)])
colorbar
figure(4)
fc_graphics4mesh.plotmesh(q2,me2,'color','LightGray','cutPlan',P)
hold on
fc_graphics4mesh.plot(q1,me1,U1,'LineWidth',2,'cutPlan',P)
axis image;axis off
caxis([min(U2),max(U2)])
```

Listing 5: Using fc_graphics4mesh.**plot** function with a 3Ds mesh

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),'cutPlan',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 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)

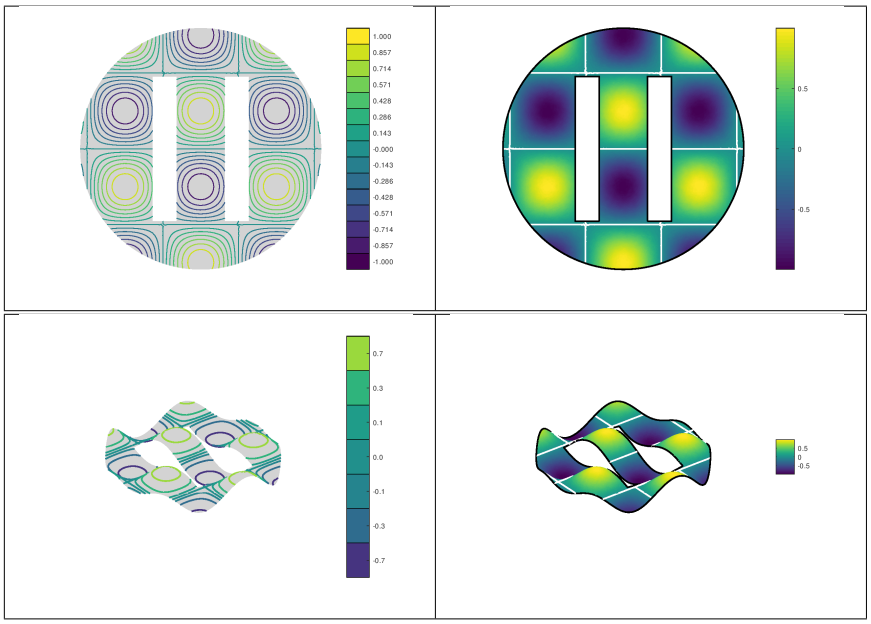
The options of second level are all options of

- **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 :

- 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.

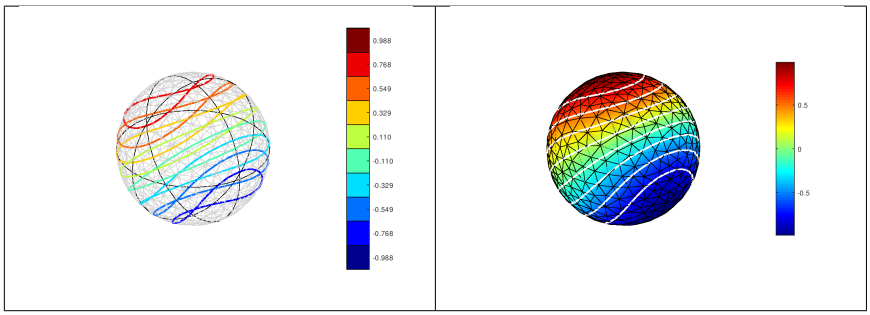
2D example : the following code is part of the `fc_graphics4mesh.demos.plotiso2D` function.



```
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,'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
```

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

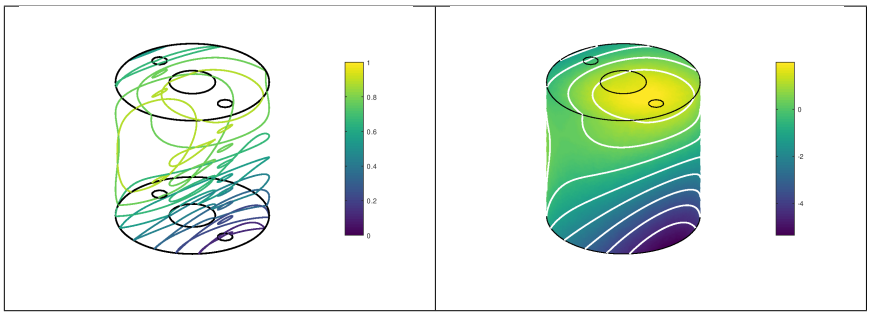
3Ds example : the following code is part of the `fc_graphics4mesh.demos.plotiso3Ds` function.




```
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)
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
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
axis image;axis off
```

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

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

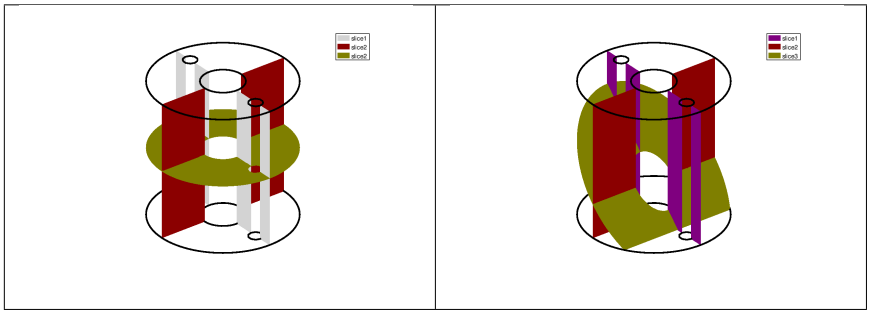
slicemesh(q,me,P) displays intersection of the plane defined by $P(1)x+P(2)y+P(3)z+P(4)=0$ and all the 3-dimensional simplices elements given by q and me arrays. To compute P one can use the fc_tools.graphics.PlaneCoefs function of the FC-TOOLS package. The 1-by-4 array P, is obtained with P=fc_tools.graphicsPlaneCoefs(Q,V) 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.

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)
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
```

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

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

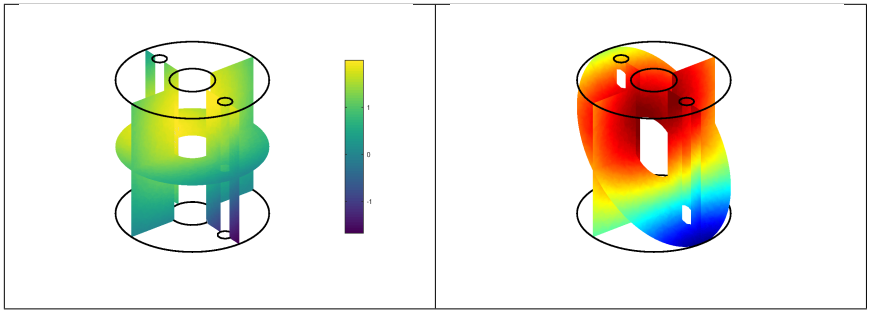
```
fc_graphics4mesh.slice(q,me,u,P)
fc_graphics4mesh.slice(q,me,u,P,Name,Value, ...)
```

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 package. The 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.

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)
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 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 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

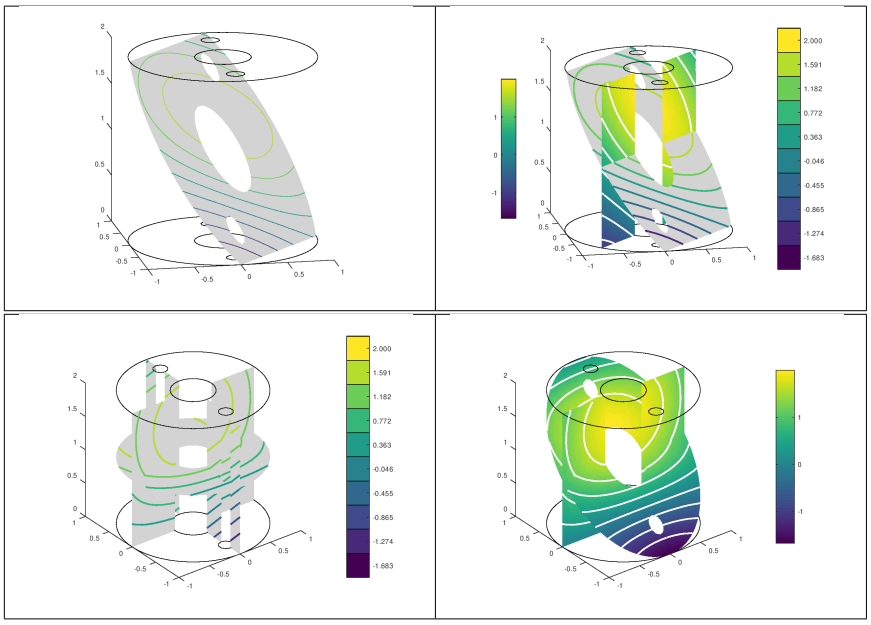
- `'niso'` : to specify the number of isolines (default : 10)
- `'isorange'` : to specify the list of isovalues (default : empty)
- `'color'` : to specify one color for all isolines (default : empty)
- `'isocolorbar'` : if true display a colorbar.Default is false.
- `'format'` : to specify the format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

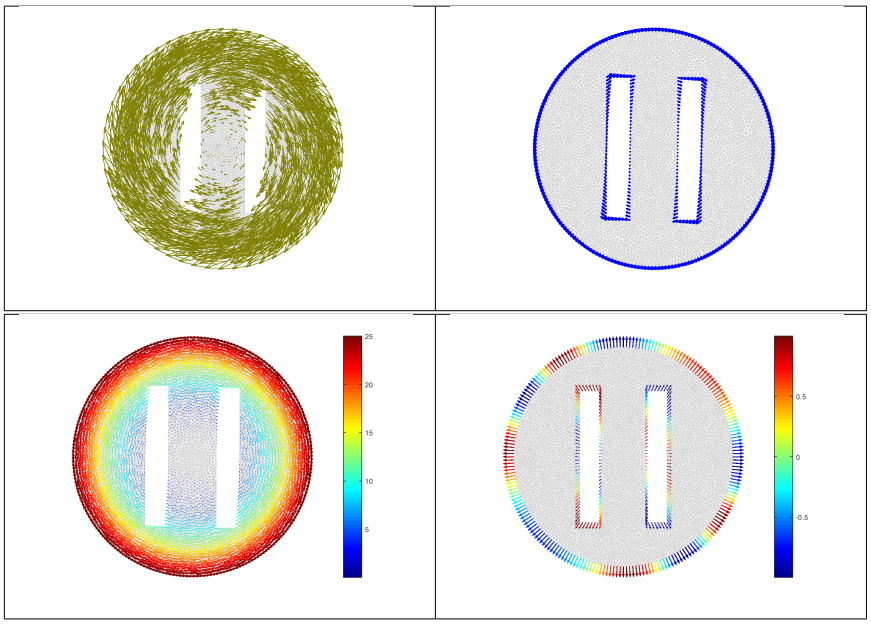
plotquiver(q,me,V,Name,Value, ...) specifies function options using one or more Name,Value pair arguments. Options of first level are

- `'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 colorized with a 1-by-nq 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.

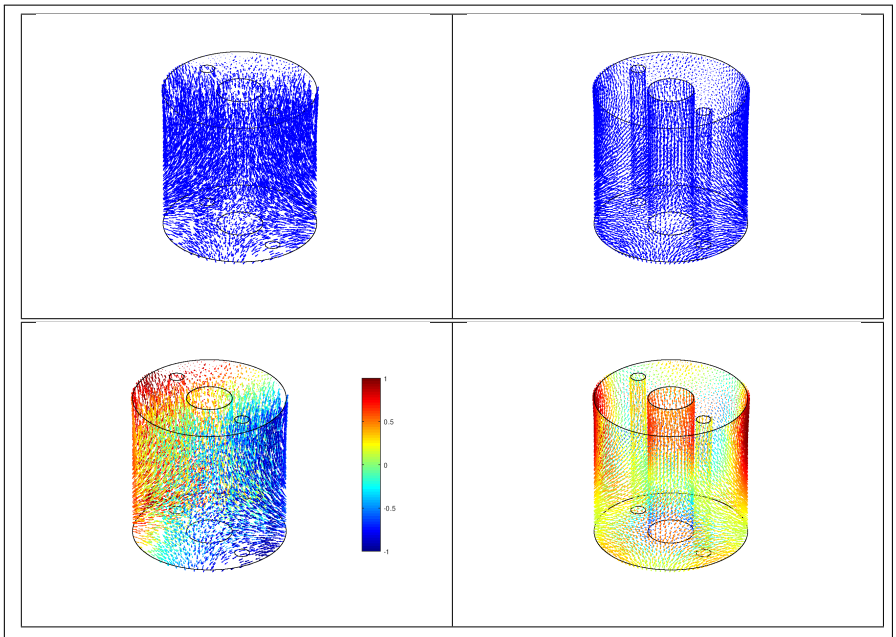
2D example



```
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

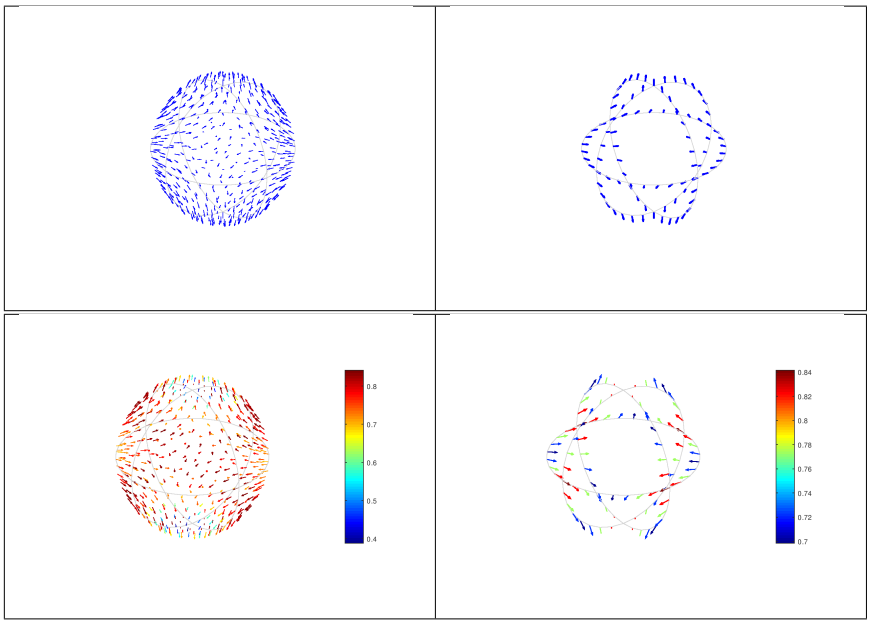
3D example



```
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 : plot quivers function

11 plotnodes function

The function 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.

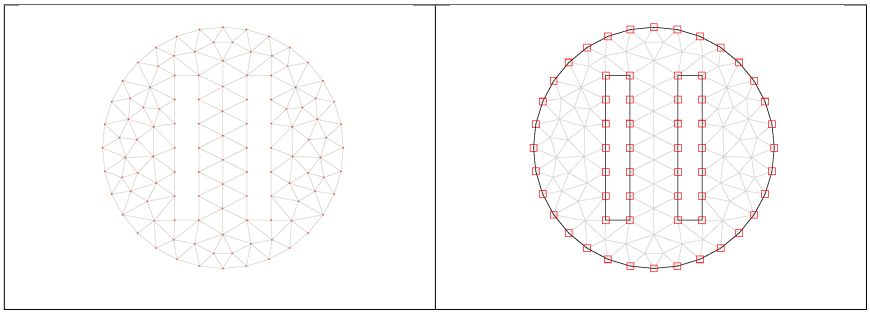
plotnodes(q,Name,Value, ...) specifies options using one or more Name,Value pair arguments. Options of first level are

- '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.

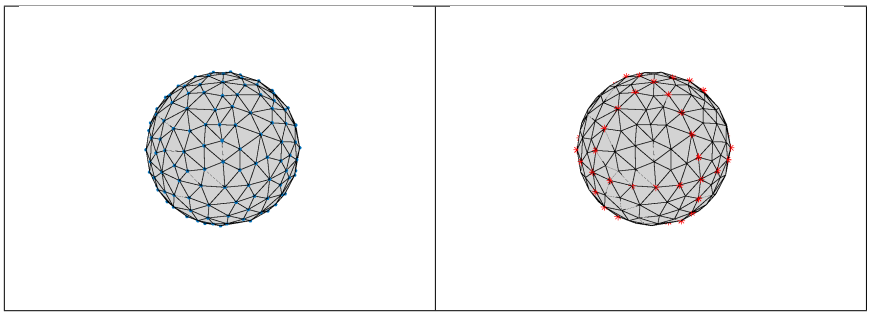
2D example : the following code is part of the `fc_graphics4mesh.demos.plotnodes2D` function.



```
[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 16: Using fc_graphics4mesh.**plotnodes** function with a 2D mesh

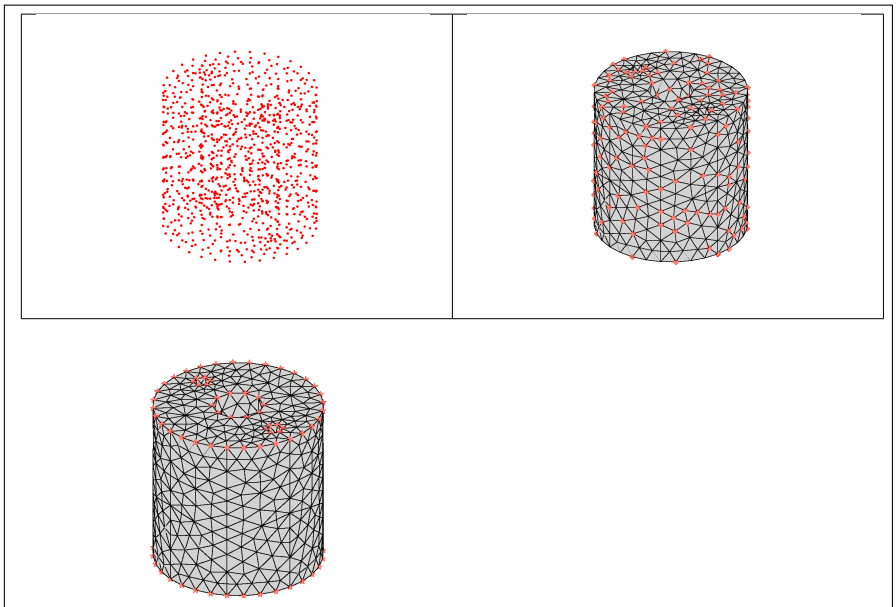
3Ds example : the following code is part of the `fc_graphics4mesh.demos.plotnodes3Ds` function.



```
[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.plotnodes(q2,'MarkerSize',15)
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','*')
```

Listing 17: Using fc_graphics4mesh.**plotnodes** function with a 3Ds mesh

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)
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')
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 18: Using fc_graphics4mesh.**plotnodes** function with a 3D mesh

12 plotnodesidx function

The function PLOTNODESIDX displays indices of the given mesh nodes array

Syntaxe

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

Description

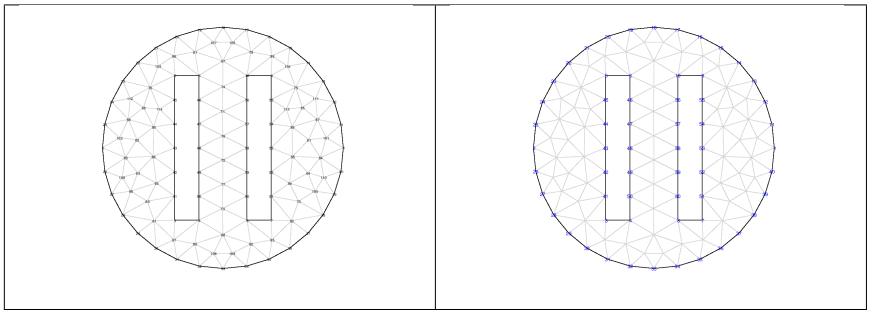
plotnodesidx(q) displays all the numbers/indices of the nodes array q

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.

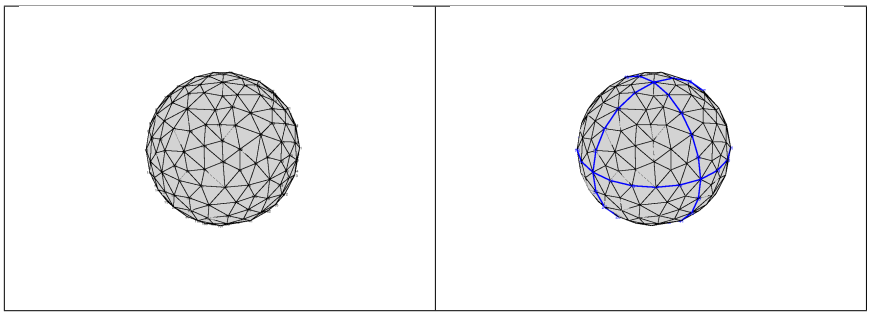
2D example : the following code is part of the `fc_graphics4mesh.demos.plotnodesidx2D` function.



```
[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

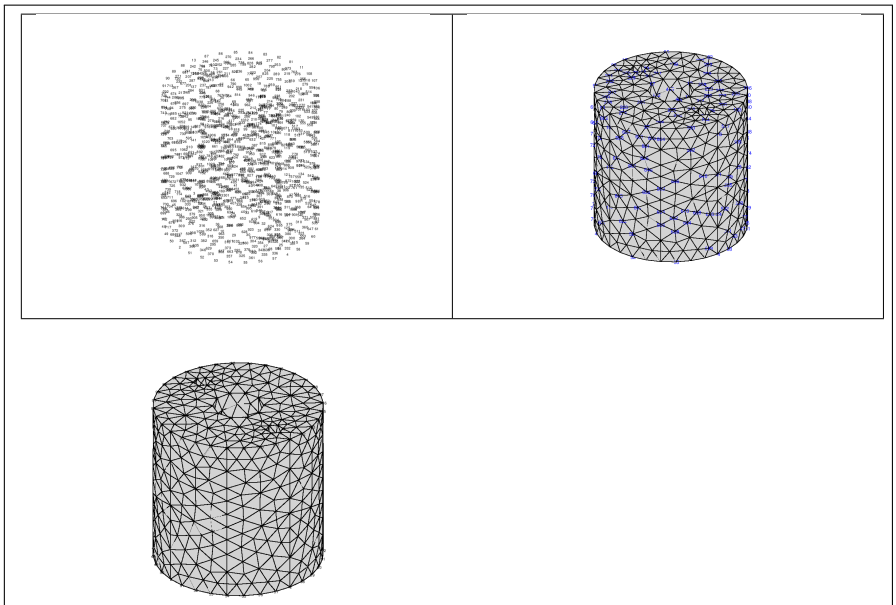
3Ds example : the following code is part of the `fc_graphics4mesh.demos.plotnodesidx3Ds` function.



```
[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.plotnodesidx(q2)
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.plotnodesidx(q1,'Color','b', 'backgroundcolor','Salmon', ...
    'edgecolor','b','fontsize',7)
```

Listing 20: Using fc_graphics4mesh.**plotnodesidx** function with a 3Ds mesh

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)
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)
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 21: Using fc_graphics4mesh.**plotnodesidx** function with a 3D mesh

13 plotelementsidx function

The function PLOTELEMENTSIDX displays indices of a given mesh connectivity array

Syntaxe

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

Description

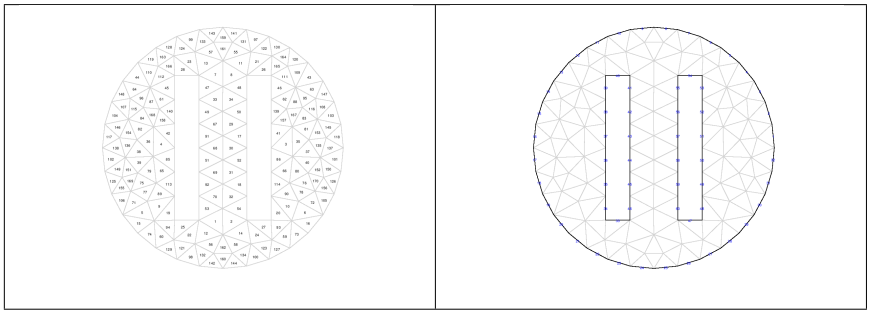
plotelementsidx(q,me) displays all the numbers/indices of the connectivity array me.

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.

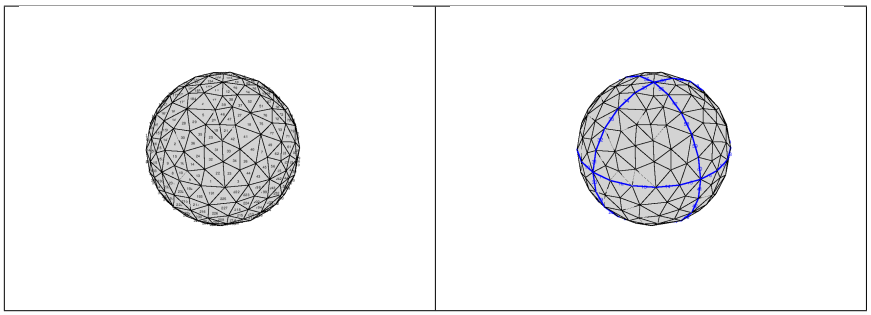
2D example : the following code is part of the `fc_graphics4mesh.demos.plotelementsidx2D` function.



```
[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 22: Using fc_graphics4mesh.**plotelementsidx** function with a 2D mesh

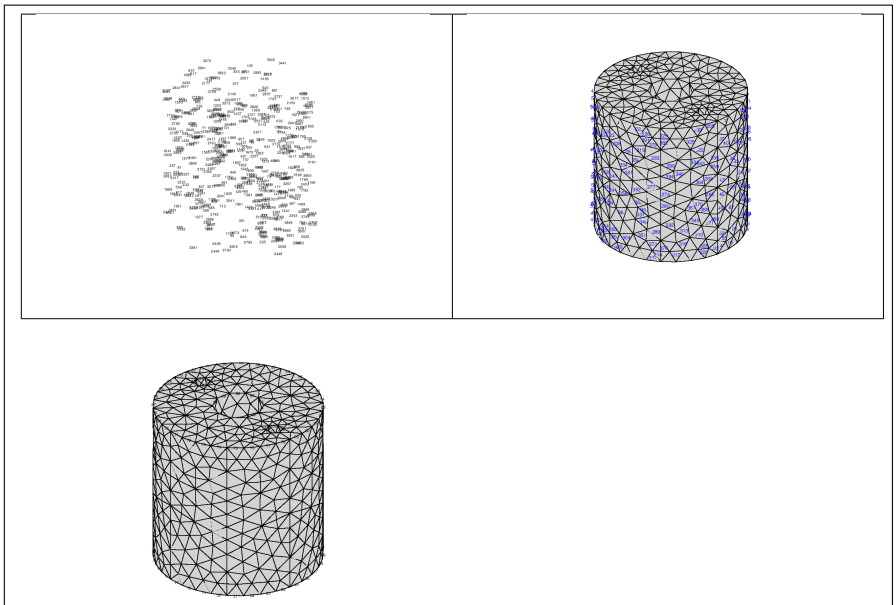
3Ds example : the following code is part of the `fc_graphics4mesh.demos.plotelementsidx3Ds` function.



```
[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)
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 23: 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)
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 graphics4mesh Octave package

git informations on the packages used to build this manual

```
-------------------------------------------------
  name : fc-graphics4mesh
   tag : 0.1.0
commit : e89efbfb480710f5e5b6c7401fc79fce1706fc24
  date : 2020-01-29
  time : 14-26-49
status : 0
-------------------------------------------------
  name : fc-tools
   tag : 0.0.29
commit : a3251e64e77572bb3916e665ae01baade6fc5f1a
  date : 2019-12-15
  time : 11-23-10
status : 0
-------------------------------------------------
  name : fc-bench
   tag : 0.1.1
commit : 9892a2270513cc1105d32235fcda24729cf53175
  date : 2019-12-18
  time : 15-46-45
status : 0
-------------------------------------------------
  name : fc-amat
   tag : 0.1.1
commit : ab639dd1d3be0ed76a8df9a9b637400ad797f32a
  date : 2020-01-02
  time : 06-40-28
status : 0
-------------------------------------------------
  name : fc-meshtools
   tag : 0.1.2
commit : 7ab773f66e612beb5203441f9d9c832d4f6e497a
  date : 2020-01-25
  time : 09-28-26
status : 0
-------------------------------------------------
```

git informations on the LaTeX package used to build this manual

```
-------------------------------------------------
  name : fctools
   tag :
commit : 7ad9c7de44262e116aa101aeae74c5e5aee6ef61
  date : 2019-10-30
  time : 13:57:21
status : 1
-------------------------------------------------
```

Using the remote configuration repository:

url	ssh://lagagit/MCS/Cuvelier/Matlab/fc-config
commit	8be709af2ac61dfa790695a47c07c905762e0440