



# Octave package, User's Guide\* version 0.4.5

François Cuvelier<sup>†</sup>

January 5, 2023

## Abstract

This object-oriented Octave package allows to use simplicial meshes generated from `gmsh` (in dimension 2 or 3) or an hypercube triangulation (in any dimension). For graphical representation the `plot` package is used.

## 0 Contents

|          |                                                           |          |
|----------|-----------------------------------------------------------|----------|
| <b>1</b> | <b>Introduction</b>                                       | <b>2</b> |
| <b>2</b> | <b>Installation</b>                                       | <b>4</b> |
| <b>3</b> | <b>Mesh Objects</b>                                       | <b>5</b> |
| 3.1      | <code>fc_simesh.siMeshElt</code> object . . . . .         | 6        |
| 3.2      | <code>fc_simesh.siMesh</code> object . . . . .            | 7        |
| 3.3      | Mesh samples . . . . .                                    | 8        |
| 3.3.1    | 2-simplicial mesh in $\mathbb{R}^2$ . . . . .             | 8        |
| 3.3.2    | Sample of a 3-simplicial mesh in $\mathbb{R}^3$ . . . . . | 10       |

\* $\text{\LaTeX}$  manual, revision 0.4.5.a, compiled with Octave 7.3.0, and packages `fc-simesh`[0.4.5], `fc-tools`[0.0.34], `fc-bench`[0.1.3], `fc-hypermesh`[1.0.4], `fc-amat`[0.1.3], `fc-meshtools`[0.1.4], `fc-graphics4mesh`[0.1.5], `fc-oogmsh`[0.2.4], `fc-siplt`[0.2.5]

<sup>†</sup>Université Sorbonne Paris Nord, LAGA, CNRS, UMR 7539, F-93430, Villetaneuse, France, [cuvelier@math.univ-paris13.fr](mailto:cuvelier@math.univ-paris13.fr).

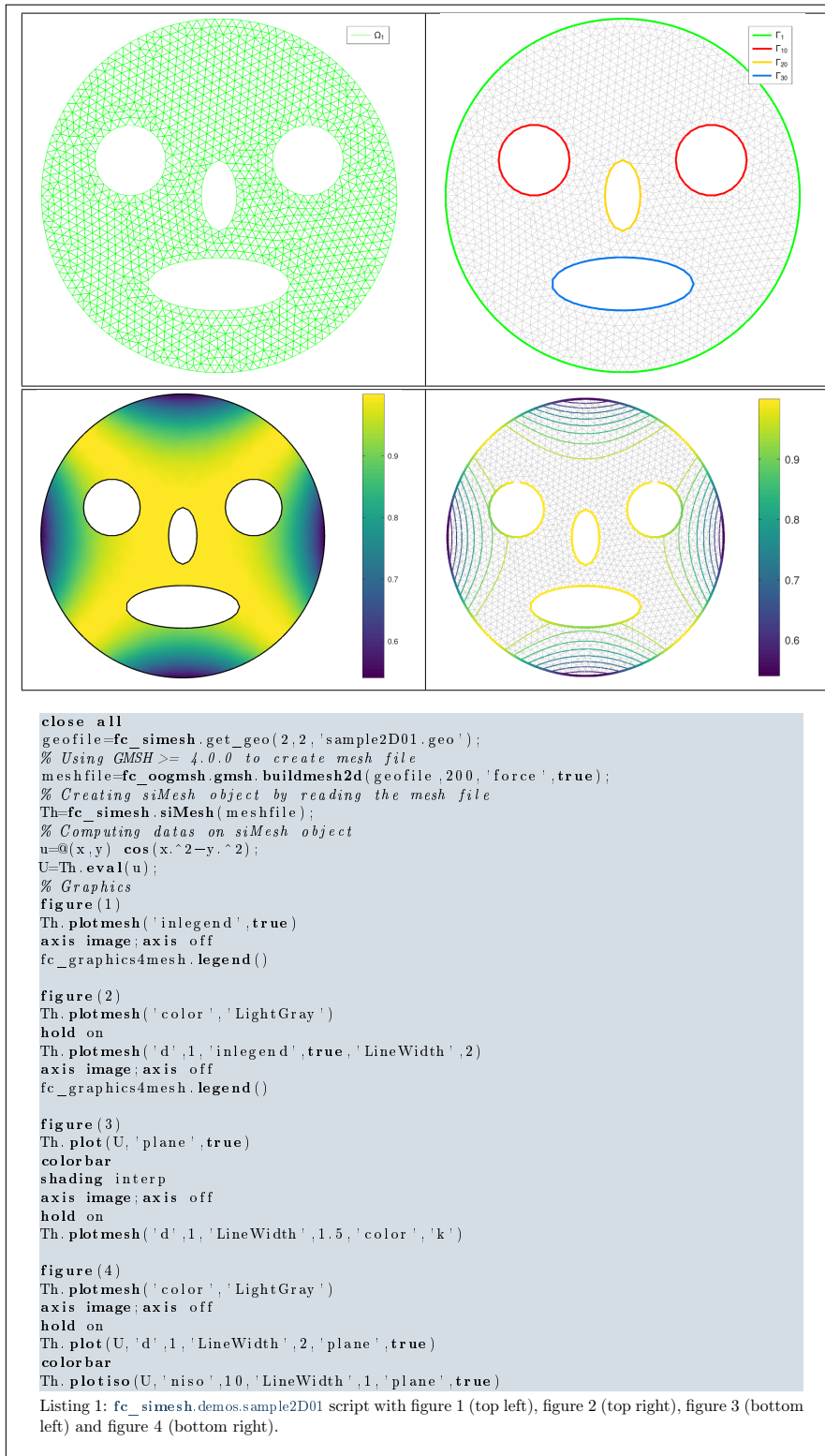
This work was supported by the ANR project DEDALES under grant ANR-14-CE23-0005.

|        |                                                                  |    |
|--------|------------------------------------------------------------------|----|
| 3.3.3  | Sample of a 2-simplicial mesh in $\mathbb{R}^3$ . . . . .        | 12 |
| 3.4    | Methods of the <code>fc_simesh.siMesh</code> object . . . . .    | 13 |
| 3.4.1  | <code>fc_simesh.siMesh</code> constructor . . . . .              | 13 |
| 3.4.2  | <code>find</code> method . . . . .                               | 14 |
| 3.4.3  | <code>feval</code> method . . . . .                              | 15 |
| 3.4.4  | <code>eval</code> method . . . . .                               | 16 |
| 3.4.5  | <code>get_h</code> method . . . . .                              | 18 |
| 3.4.6  | <code>get_mesh</code> method . . . . .                           | 18 |
| 3.4.7  | <code>get_nme</code> method . . . . .                            | 18 |
| 3.4.8  | <code>get_nq</code> method . . . . .                             | 19 |
| 3.4.9  | <code>get_labels</code> method . . . . .                         | 20 |
| 3.4.10 | <code>move</code> method . . . . .                               | 20 |
| 3.4.11 | <code>plotmesh</code> method . . . . .                           | 26 |
| 3.4.12 | <code>plot</code> method . . . . .                               | 31 |
| 3.4.13 | <code>plotiso</code> method . . . . .                            | 34 |
| 3.4.14 | <code>slicemesh</code> method . . . . .                          | 38 |
| 3.4.15 | <code>slice</code> method . . . . .                              | 39 |
| 3.4.16 | <code>sliceiso</code> method . . . . .                           | 40 |
| 3.4.17 | <code>plotquiver</code> method . . . . .                         | 42 |
| 3.4.18 | <code>scatter</code> method . . . . .                            | 46 |
| 3.5    | Hypercube as a <code>fc_simesh.siMesh</code> object . . . . .    | 50 |
| 3.5.1  | 2D hypercube . . . . .                                           | 51 |
| 3.5.2  | 3D hypercube . . . . .                                           | 51 |
| 3.5.3  | 4D hypercube . . . . .                                           | 52 |
| 3.5.4  | 5D hypercube . . . . .                                           | 53 |
| 3.5.5  | Möbius strip as a <code>fc_simesh.siMesh</code> object . . . . . | 53 |

## Appendices 56

### 1 Introduction

The Octave package was created to simplify the use of simplicial meshes and to easily represent data on all or parts of them. In 2D or 3D `gmsh` can be used under Octave to build triangular or tetrahedral meshes by using the `package[1]`. Thereafter the mesh stored as a file (`.msh`) can be read by using the `fc_simesh.siMesh` object. In Listing 1, a 2D example is provided with the 4 generated figures. For graphic representations, the `package[2]` is used by the `fc_simesh.siMesh` object `Th` as follows `Th.plotmesh(...)` is `fc_siplt.plotmesh(Th,...)`, `Th.plot(...)` is `fc_siplt.plot(Th,...)` and so on.



In higher dimension the `package[3]` can be used to obtain meshes of an hypercube by using the `fc_simesh.hypercube` function.

## 2 Installation

One just has to get/download the install file

`ofc_simesh_install.m`

or get it on the dedicated web page. Thereafter, one run it under Octave. This command download, extract and configure the [fc\\_simesh](#) packages and all the required packages in the current directory.

For example, to install this package in `~/Octave` directory, one has to copy the file `ofc_simesh_install.m` in the `~/Octave` directory. Then in a Octave terminal run the following commands to install the *fc-simesh* package with graphical extension

```
>> cd ~/Octave
>> ofc_simesh_install
```

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

```
Parts of the <fc-simesh> Octave package.
Copyright (C) 2017-2022 F. Cuvelier

1- Downloading and extracting the packages
2- Setting the <fc-simesh> package
Write in ~/Octave/fc-simesh-full/fc_simesh-0.4.5/configure_loc.m ...
3- Using packages :
->          fc-tools : 0.0.34
->          fc-bench : 0.1.3
->          fc-hypermesh : 1.0.4
->          fc-amat : 0.1.3
->          fc-meshtools : 0.1.4
->          fc-graphics4mesh : 0.1.5
->          fc-oogmsh : 0.2.4
->          fc-siplt : 0.2.5
with          fc-simesh : 0.4.5
*** Using instructions
To use the <fc-simesh> package:
addpath('~/Octave/fc-simesh-full/fc_simesh-0.4.5')
fc_simesh.init()

See ~/Octave/ofc_simesh_set.m
```

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

```
>> addpath('~/Octave/fc-simesh-full/fc_simesh-0.4.5')
>> fc_simesh.init()
```

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



```

Try to use default parameters!
Use fc_tools.configure to configure.
Write in ~/Octave/fc-simesh-full/fc_tools-0.0.34/configure_loc.m ...
Try to use default parameters!
Use fc_bench.configure to configure.
Write in ~/Octave/fc-simesh-full/fc_bench-0.1.3/configure_loc.m ...
Try to use default parameters!
Use fc_hypermesh.configure to configure.
Write in ~/Octave/fc-simesh-full/fc_hypermesh-1.0.4/configure_loc.m ...
Try to use default parameters!
Use fc_amat.configure to configure.
Write in ~/Octave/fc-simesh-full/fc_amat-0.1.3/configure_loc.m ...
Try to use default parameters!
Use fc_meshtools.configure to configure.
Write in ~/Octave/fc-simesh-full/fc_meshtools-0.1.4/configure_loc.m ...
Try to use default parameters!
Use fc_graphics4mesh.configure to configure.
Write in ...
~/Octave/fc-simesh-full/fc_graphics4mesh-0.1.5/configure_loc.m ...
Try to use default parameters!
Use fc_oogmsh.configure to configure.
Write in ~/Octave/fc-simesh-full/fc_oogmsh-0.2.4/configure_loc.m ...
Try to use default parameters!
Use fc_siplt.configure to configure.
Write in ~/Octave/fc-simesh-full/fc_siplt-0.2.5/configure_loc.m ...
Using fc_simesh[0.4.5] with fc_tools[0.0.34], fc_bench[0.1.3], ...
    fc_hypermesh[1.0.4],
                                fc_amat[0.1.3], fc_meshtools[0.1.4], ...
                                fc_graphics4mesh[0.1.5],
                                fc_oogmsh[0.2.4], fc_siplt[0.2.5].
[fc-oogmsh] Configured to use gmsh 4.11.0 with default MSH file format ...
version 4.1

```

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

```

CUUsing fc_simesh[0.4.5] with fc_tools[0.0.34], fc_bench[0.1.3], ...
    fc_hypermesh[1.0.4],
                                fc_amat[0.1.3], fc_meshtools[0.1.4], ...
                                fc_graphics4mesh[0.1.5],
                                fc_oogmsh[0.2.4], fc_siplt[0.2.5].
[fc-oogmsh] Configured to use gmsh 4.11.0 with default MSH file format ...
version 4.1

```

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

~/Octave/fc-simesh-full

### 3 Mesh Objects

In geometry, a simplex is a generalization of the notion of a triangle or tetrahedron to arbitrary dimensions. Specifically, a  $k$ -simplex in  $\mathbb{R}^{\dim}$ ,  $0 \leq k \leq \dim$ , is a polytope which is the convex hull of its  $k + 1$  vertices of  $\mathbb{R}^{\dim}$ . More formally, suppose the  $k + 1$  vertices  $q^0, \dots, q^k \in \mathbb{R}^{\dim}$  such that  $q^1 - q^0, \dots, q^k - q^0$  are linearly independent. Then, the  $k$ -simplex  $K$  determined by them is the set of points

$$K = \left\{ \sum_{i=0}^k \lambda_i q^i \mid \lambda_i \geq 0, i \in \llbracket 0, k \rrbracket, \text{ with } \sum_{i=0}^k \lambda_i = 1 \right\}.$$

We denote by  **$k$ -simplicial elementary mesh** in  $\mathbb{R}^{\dim}$ ,  $0 \leq k \leq \dim$ , a mesh with **unique label** only composed with  $k$ -simplices.

A  **$d$ -simplicial mesh** in  $\mathbb{R}^{\dim}$ ,  $0 \leq d \leq \dim$ , is an union of  $k$ -simplicial elementary meshes with  $k \in \llbracket 0, d \rrbracket$ .

### 3.1 `fc_simesh.siMeshElt` object

An elementary  $d$ -simplicial mesh in dimension `dim` is represented by the class `fc_simesh.siMeshElt`. We give properties of this class :



#### Properties of `fc_simesh.siMeshElt` object for $d$ -simplicial elementary meshes in $\mathbb{R}^{\dim}$

|                        |                                                                                                                                                                                                                                                                                                                                                                                                  |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>dim</code>       | : integer<br>space dimension                                                                                                                                                                                                                                                                                                                                                                     |
| <code>d</code>         | : integer ( $0 \leq d \leq \dim$ )                                                                                                                                                                                                                                                                                                                                                               |
| <code>nq</code>        | : integer<br>number of vertices                                                                                                                                                                                                                                                                                                                                                                  |
| <code>nme</code>       | : integer<br>number of elements ( $d$ -simplices )                                                                                                                                                                                                                                                                                                                                               |
| <code>q</code>         | : $\dim$ -by- <code>nq</code> array of reals<br>array of vertex coordinates                                                                                                                                                                                                                                                                                                                      |
| <code>me</code>        | : $(d + 1)$ -by- <code>nme</code> array of integers<br>connectivity array for <b>mesh elements</b>                                                                                                                                                                                                                                                                                               |
| <code>vols</code>      | : 1-by- <code>nme</code> array of reals<br>array of mesh element volumes                                                                                                                                                                                                                                                                                                                         |
| <code>h</code>         | : double<br>mesh step size (=maximum edge length in the mesh)                                                                                                                                                                                                                                                                                                                                    |
| <code>toGlobal</code>  | : 1-by- <code>nq</code> array of integers<br>convert from local to global mesh vertices numbering.<br>Prefer the use of <code>toParents{end}</code> instead.<br><i>It will be removed in a future release.</i>                                                                                                                                                                                   |
| <code>toParent</code>  | : 1-by- <code>nq</code> array of integers<br>convert from local to parent mesh vertices numbering<br>(same as <code>toGlobal</code> if not part of a partitioned mesh).<br>Prefer the use of <code>toParents{1}</code> instead.<br><i>It will be removed in a future release.</i>                                                                                                                |
| <code>nqParents</code> | : 1-by- $n$ array of integers<br><code>nqParents(1)</code> number of vertices in the <i>parent</i> mesh,<br><code>nqParents(2)</code> number of vertices in the <i>parent</i> of the <i>parent</i> mesh,<br><code>nqParents(end)</code> number of vertices in the global mesh.                                                                                                                   |
| <code>toParents</code> | : 1-by- $n$ cell array<br><code>toParents{1}</code> indices array which convert local vertices numbering to the <i>parent</i> mesh vertices numbering,<br><code>toParents{2}</code> indices array which convert local vertices numbering to the <i>parent</i> of the <i>parent</i> mesh,<br><code>toParents{end}</code> indices array which convert local vertices numbering to the global mesh. |

More precisely

- $q(i,j)$  is the  $i$ -th coordinate of the  $j$ -th vertex,  $i \in \{1, \dots, \dim\}$ ,  $j \in \{1, \dots, nq\}$ . The  $j$ -th vertex will be also denoted by  $q^j = q(:, j)$ .

- `me(r,k)` is the storage index of the  $r$ -th vertex of the  $k$ -th element ( $d$ -simplex), in the array `q`, for  $r \in \{1, \dots, d+1\}$  and  $k \in \{1, \dots, nme\}$ . So `q(:, me(r,k))` represents the coordinates of the  $r$ -th vertex of the  $k$ -th mesh element.
- `vols(k)` is the volume of the  $k$ -th  $d$ -simplex.

### 3.2 `fc_simesh.siMesh` object

A  $d$ -simplicial mesh in dimension `dim`, represented as an `fc_simesh.siMesh` object, is an union of `fc_simesh.siMeshElt` objects which are elementary  $l$ -simplicial meshes ( $0 \leq l \leq d$ ) in space dimension `dim`.



#### `fc_simesh.siMesh` object properties

|                        |                                                                                                                                                                                                                                                                              |
|------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>dim</code>       | : integer<br>space dimension                                                                                                                                                                                                                                                 |
| <code>d</code>         | : integer<br>$d$ -dimensional simplicial mesh                                                                                                                                                                                                                                |
| <code>sTh</code>       | : array of <code>fc_simesh.siMeshElt</code> objects                                                                                                                                                                                                                          |
| <code>nsTh</code>      | : number of <code>fc_simesh.siMeshElt</code> objects                                                                                                                                                                                                                         |
| <code>sThsimp</code>   | : array of <code>nsTh</code> integers<br>$i$ -th <code>fc_simesh.siMeshElt</code> object in <code>sTh</code> is a <code>sThsimp(i)</code> -simplicial elementary mesh                                                                                                        |
| <code>sThlab</code>    | : array of <code>nsTh</code> integers in <code>sTh</code><br>label of $i$ -th <code>fc_simesh.siMeshElt</code> object in <code>sTh</code> is number <code>sThlab(i)</code>                                                                                                   |
| <code>nq</code>        | : integer<br>number of vertices in the mesh                                                                                                                                                                                                                                  |
| <code>toGlobal</code>  | : 1-by- <code>nq</code> array of integers<br>convert from local to global mesh vertices numbering.<br>Prefer the use of <code>ndtoParent</code> instead.<br><i>It will be removed in a future release.</i>                                                                   |
| <code>toParent</code>  | : 1-by- <code>nq</code> array of integers<br>convert from local to parent mesh vertices numbering (same as <code>toGlobal</code> if not part of a partitioned mesh).<br>Prefer the use of <code>toParents1</code> instead.<br><i>It will be removed in a future release.</i> |
| <code>nqParents</code> | : 1-by- $n$ array of integers<br>Only used with partitioned mesh and the <b>FC-PSIMESH</b> package.                                                                                                                                                                          |
| <code>toParents</code> | : 1-by- $n$ cell array<br>Only used with partitioned mesh and the <b>FC-PSIMESH</b> package.                                                                                                                                                                                 |

Let `Th` be a `fc_simesh.siMesh` object. The global `dim`-by-`(Th.nq)` array `q` of mesh vertices is not explicitly stored in `Th`, however one can easily build it if necessary:

```
q=zeros(Th.dim,Th.nq);
for i=Th.find(Th.d)
```

```

q(:, Th.sTh{i}).toParents{1})=Th.sTh{i}.q;
end

```

### 3.3 Mesh samples

#### 3.3.1 2-simplicial mesh in $\mathbb{R}^2$

Listing 2: 2D `fc_simesh.siMesh` object from `sample20.geo`

```

meshfile=fc_oogmsh.gmsh.buildmesh2d('sample20',20,'force',false);
Th=fc_simesh.siMesh(meshfile);
fprintf('***_Th:\n')
disp(Th)
fprintf('***_Th.sTh{9}:\n')
disp(Th.sTh{9})

```

Output

```

[fc-oogmsh] Input file : <fc-oogmsh>/geodir/2d/sample20.geo
[fc-oogmsh] Starting building mesh <fc-oogmsh>/meshes/sample20-20.msh with gmsh 4.11.0
[fc-oogmsh] Using command : gmsh -2 -setnumber N 20 -string "Mesh.MshFileVersion=4.1;" ...
<fc-oogmsh>/geodir/2d/sample20.geo -o <fc-oogmsh>/meshes/sample20-20.msh
Be patient...
[fc-oogmsh] Using gmsh 4.11.0 to write MSH file format version 4.1 in ...
<fc-oogmsh>/meshes/sample20-20.msh
*** Th:
fc_simesh.siMesh with properties:
  bbox: [ -1 1 -1 1 ] (1x4 double)
  d: 2 double
  dim: 2 double
  nq: 2327 double
  nqParents: 2327 double
  nsTh: 11 double
  other: (1x1 struct)
  sTh: (1x11 cell)
sThPhysicalTags: []
sThboundlabs: []
sThcolors: (11x3 double)
sThgeolab: []
sThlab: [ 1 2 10 20 1 2 20 101 102 103 104 ] (1x11 double)
sThpartlabs: []
sThphyslab: [ 1 2 10 20 1 2 20 101 102 103 104 ] (1x11 double)
sThsimp: [ 2 2 2 2 1 1 1 1 1 1 1 ] (1x11 double)
toGlobal: (1x2327 double)
toParent: (1x2327 double)
toParents: (1x1 cell)
*** Th.sTh{9}:
siMeshElt with properties:
  Tag: (1x30 char)
  bbox: [ 1 1 -1 1 ] (1x4 double)
  color: [ 0.0344828 0.448276 0.0689655 ] (1x3 double)
  d: 1 double
  dim: 2 double
  geolab: []
  gradBaCo: (40x2x2 double)
  h: 0.05 double
  label: 102 double
  me: (2x40 double)
  nme: 40 double
  normals: (2x40 double)
  nq: 41 double
  nqGlobal: 2327 double
  nqParent: 2327 double
  nqParents: 2327 double
  order: 1 double
  partlab: []
  q: (2x41 double)
  toGlobal: (1x41 double)
  toParent: (1x41 double)
  toParents: (1x1 cell)
  vols: (1x40 double)
[fc-tools] waiting 2(s) to finish saving figures

```

From the output of the Listing 2 or from the Figure 1 the complete domain

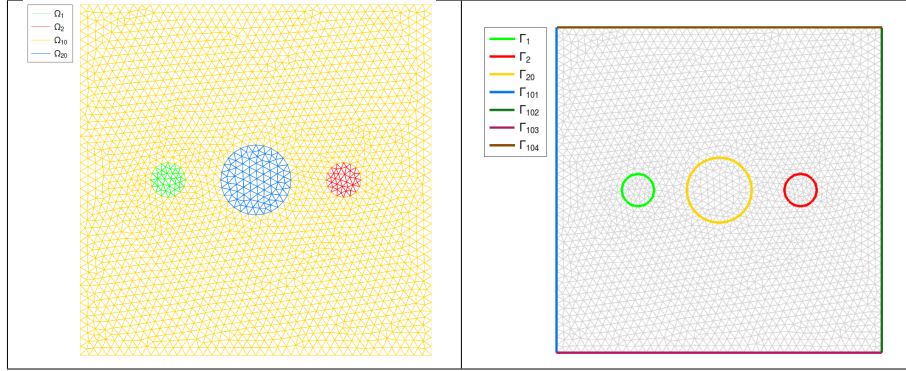


Figure 1: 2D `fc_simesh.siMesh` object from `sample20.geo`

is

$$\Omega = \Omega_1 \cup \Omega_2 \cup \Omega_{10} \cup \Omega_{20}$$

and we note

$$\Gamma = \Gamma_1 \cup \Gamma_2 \cup \Gamma_{20} \cup \Gamma_{101} \cup \Gamma_{102} \cup \Gamma_{103} \cup \Gamma_{104}.$$

So this mesh is 2-simplicial mesh in  $\mathbb{R}^2$  and is composed of :

- four 2-simplicial elementary meshes :  $\Omega_i, \forall i \in \{1, 2, 10, 20\}$
- seven 1-simplicial elementary meshes :  $\Gamma_i \forall i \in \{1, 2, 20, 101, 102, 104\}$

### 3.3.2 Sample of a 3-simplicial mesh in $\mathbb{R}^3$

Listing 3: : 3D Mesh from quart\_sphere2.geo

```
meshfile=fc_oogmsh.gmsh.buildmesh3d('quart_sphere2',5);
Th=fc_simesh.siMesh(meshfile);
fprintf('***_Th:\n')
disp(Th)
fprintf('***_Th.sTh{9}:\n')
disp(Th.sTh{9})
```

#### Output

```
[fc-oogmsh] Input file : <fc-oogmsh>/geodir/3d/quart_sphere2.geo
[fc-oogmsh] Starting building mesh <fc-oogmsh>/meshes/quart_sphere2-5.msh with gmsh 4.11.0
[fc-oogmsh] Using command : gmsh -3 -setnumber N 5 -string "Mesh.MshFileVersion=4.1;" ...
<fc-oogmsh>/geodir/3d/quart_sphere2.geo -o <fc-oogmsh>/meshes/quart_sphere2-5.msh
Be patient...
[fc-oogmsh] Using gmsh 4.11.0 to write MSH file format version 4.1 in ...
<fc-oogmsh>/meshes/quart_sphere2-5.msh
Mesh quart_sphere2-5.msh is a 3-dimensional mesh
Force dimension to 3
*** Th:
fc_simesh.siMesh with properties:
    bbox: [ -1 1 0 1 0 1 ] (1x6 double)
    d: 3 double
    dim: 3 double
    nq: 1172 double
    nqParents: 1172 double
    nsTh: 23 double
    other: (1x1 struct)
    sTh: (1x23 cell)
    sThPhysicalTags: []
    sThboundlabs: []
    sThcolors: (23x3 double)
    sThgeolab: []
    sThlab: [ 1 2 1 2 3 4 5 6 7 1 2 3 4 5 6 7 8 9 1 2 3 4 5 ] (1x23 double)
    sThpartlabs: []
    sThphyslab: [ 1 2 1 2 3 4 5 6 7 1 2 3 4 5 6 7 8 9 1 2 3 4 5 ] (1x23 double)
    sThsimp: [ 3 3 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 0 0 0 0 ] (1x23 double)
    toGlobal: (1x1172 double)
    toParent: (1x1172 double)
    toParents: (1x1 cell)
*** Th.sTh{9}:
siMeshElt with properties:
    Tag: (1x30 char)
    bbox: [ 0 1 0 1 0 1 ] (1x6 double)
    color: [ 0.517241 0.310345 0 ] (1x3 double)
    d: 2 double
    dim: 3 double
    geolab: []
    gradBaCo: (359x3x3 double)
    h: 0.12811 double
    label: 7 double
    me: (3x359 double)
    nme: 359 double
    normals: (3x359 double)
    nq: 203 double
    nqGlobal: 1172 double
    nqParent: 1172 double
    nqParents: 1172 double
    order: 1 double
    partlab: []
    q: (3x203 double)
    toGlobal: (1x203 double)
    toParent: (1x203 double)
    toParents: (1x1 cell)
    vols: (1x359 double)
[fc-tools] waiting 2(s) to finish saving figures
[fc-tools] waiting 2(s) to finish saving figures
```

The mesh obtained from Listing 3 is a 3-simplicial mesh in  $\mathbb{R}^3$  and is composed of :

- two 3-simplicial elementary meshes :  $\Omega_i, \forall i \in \{1, 2\}$
- seven 2-simplicial elementary meshes :  $\Gamma_i \forall i \in \llbracket 1, 7 \rrbracket$

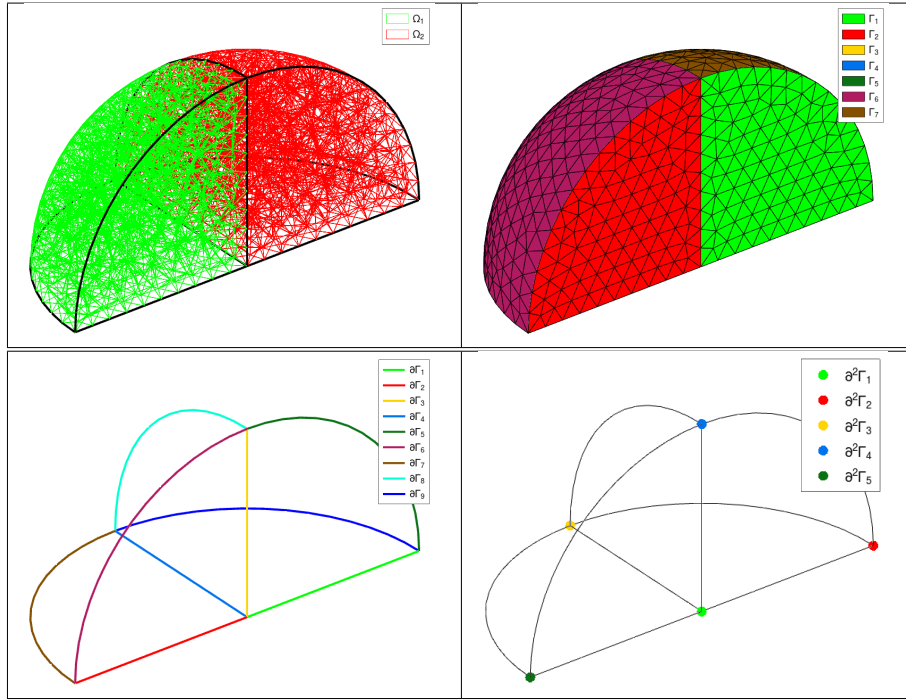


Figure 2: 3D Mesh from `quart_sphere2.geo`

- nine 1-simplicial elementary meshes :  $\partial\Gamma_i \ \forall i \in \llbracket 1, 9 \rrbracket$
- five 0-simplicial elementary meshes :  $\partial^2\Gamma_i \ \forall i \in \llbracket 1, 5 \rrbracket$

### 3.3.3 Sample of a 2-simplicial mesh in $\mathbb{R}^3$

Listing 4: 3D surface Mesh from demisphere4surf.geo

```
meshfile=fc_oogmsh.gmsh.buildmesh3ds('demicircle4surf',5,'force',true);
Th=fc_simesh.siMesh(meshfile);
fprintf('***_Th:\n')
disp(Th)
fprintf('***_Th.sTh{9}:\n')
disp(Th.sTh{9})
```

#### Output

```
[fc-oogmsh] Input file : <fc-oogmsh>/geodir/3ds/demicircle4surf.geo
[fc-oogmsh] Starting building mesh <fc-oogmsh>/meshes/demicircle4surf-5.msh with gmsh 4.11.0
[fc-oogmsh] Using command : gmsh -2 -setnumber N 5 -string "Mesh.MshFileVersion=4.1;" ...
<fc-oogmsh>/geodir/3ds/demicircle4surf.geo -o <fc-oogmsh>/meshes/demicircle4surf-5.msh
Be patient...
[fc-oogmsh] Using gmsh 4.11.0 to write MSH file format version 4.1 in ...
<fc-oogmsh>/meshes/demicircle4surf-5.msh
Mesh demisphere4surf-5.msh is a 3-dimensional mesh
Force dimension to 3
*** Th:
fc_simesh.siMesh with properties:
    bbox: [ -1 1 -1 1 0 1 ] (1x6 double)
    d: 2 double
    dim: 3 double
    nq: 228 double
    nqParents: 228 double
    nsTh: 12 double
    other: (1x1 struct)
    sTh: (1x12 cell)
    sThPhysicalTags: []
    sThboundlabs: []
    sThcolors: (12x3 double)
    sThgeolab: []
    sThlab: [ 1 2 3 4 1 2 3 4 5 6 7 8 ] (1x12 double)
    sThpartlabs: []
    sThphyslab: [ 1 2 3 4 1 2 3 4 5 6 7 8 ] (1x12 double)
    sThsimp: [ 2 2 2 2 1 1 1 1 1 1 1 1 ] (1x12 double)
    toGlobal: (1x228 double)
    toParent: (1x228 double)
    toParents: (1x1 cell)
*** Th.sTh{9}:
siMeshElt with properties:
    Tag: (1x26 char)
    bbox: [ -1 0 0 0 0 1 ] (1x6 double)
    color: [ 0.0344828 0.448276 0.0689655 ] (1x3 double)
    d: 1 double
    dim: 3 double
    geolab: []
    gradBaCo: (8x2x3 double)
    h: 0.196034 double
    label: 5 double
    me: (2x8 double)
    nme: 8 double
    normals: (3x8 double)
    nq: 9 double
    nqGlobal: 228 double
    nqParent: 228 double
    nqParents: 228 double
    order: 1 double
    partlab: []
    q: (3x9 double)
    toGlobal: (1x9 double)
    toParent: (1x9 double)
    toParents: (1x1 cell)
    vols: [ 0.196034 0.196034 0.196034 0.196034 0.196034 0.196034 0.196034 0.196034 ] ...
    (1x8 double)
[fc-tools] waiting 2(s) to finish saving figures
```

The mesh obtained from the Listing 4 or from the Figure 3 is a 2-simplicial mesh in  $\mathbb{R}^3$  and is composed of :

- four 2-simplicial elementary meshes :  $\Omega_i, \forall i \in \llbracket 1, 4 \rrbracket$
- eight 1-simplicial elementary meshes :  $\Gamma_i \forall i \in \llbracket 1, 8 \rrbracket$



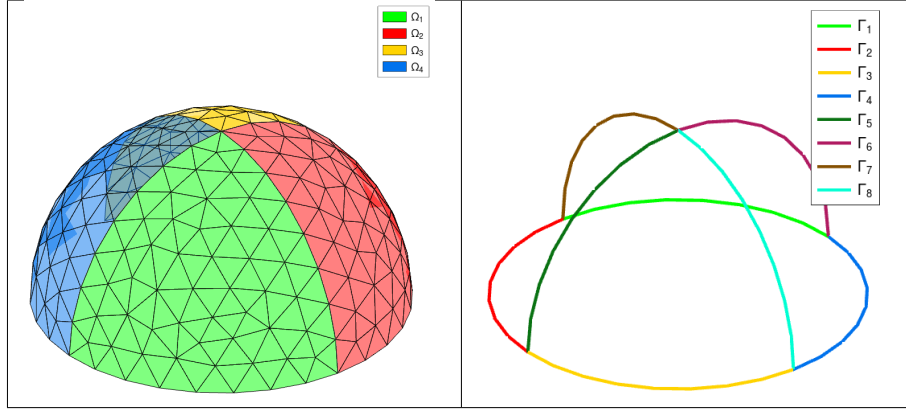


Figure 3: 3D surface Mesh from `demisphere4surf.geo`, label of the domains (left) and label of the boundaries (right)

### 3.4 Methods of the `fc_simesh.siMesh` object

#### 3.4.1 `fc_simesh.siMesh` constructor

The constructor of the `fc_simesh.siMesh` class can initialize the object from various kind of mesh file format : `.msh` (default `gmsh` format), `.mesh` (`FreeFEM++` or `Medit`) or ... (`triangle`).

##### Syntaxe

```
Th=fc_simesh.siMesh( meshfile )
Th=fc_simesh.siMesh( meshfile ,Name, Value )
```

##### Description

`Th=fc_simesh.siMesh(meshfile)` create the `fc_simesh.siMesh` object  $\mathcal{T}_h$  from the mesh file `meshfile` (`gmsh` format by default).

`Th=fc_simesh.siMesh(meshfile,Key,Value, ...)` specifies function options using one or more `Key,Value` pair arguments. The string `Key` options can be

- `'format'` : to specify the format of the mesh file `meshfile`. `Value` must be `'medit'`, `'gmsh'` (default), `'freefem'` or `'triangle'`.
- `'dim'` : to specify the space dimension (default 2),
- `'d'` : to specify the dimensions of the simplices to read, (default `[dim,dim-1]`)

**Examples** The following example use the function `fc_oogmsh.gmsh.buildmesh2d` of the `FC-OOGMSH` package to build the mesh from the `.geo` file `condenser11.geo`. This `.geo` file is located in the directory `geodir/2d` of the `FC-OOGMSH` package.

Listing 5: : `fc_simesh.siMesh` constructor

```
meshfile=fc_oogmsh.gmsh.buildmesh2d('condenser11',25,'verbose',0);
disp('***_Read_mesh_***')
Th=fc_simesh.siMesh(meshfile)
```

#### Output

```
*** Read mesh ***
Th =

    fc_simesh.siMesh with properties:
        bbox: [ -1 1 -1 1 ] (1x4 double)
        d: 2 double
        dim: 2 double
        nq: 3162 double
        nqParents: 3162 double
        nsTh: 19 double
        other: (1x1 struct)
        sTh: (1x19 cell)
        sThPhysicalTags: []
        sThboundlabs: []
        sThcolors: (19x3 double)
        sThgeolab: []
        sThlab: [ 2 4 6 8 10 20 1 2 3 4 5 6 7 8 20 101 102 103 104 ] (1x19 double)
        sThpartlabs: []
        sThphyslab: [ 2 4 6 8 10 20 1 2 3 4 5 6 7 8 20 101 102 103 104 ] (1x19 double)
        sThsimp: [ 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 ] (1x19 double)
        toGlobal: (1x3162 double)
        toParent: (1x3162 double)
        toParents: (1x1 cell)
```

### 3.4.2 find method

We denote by `Th` a `fc_simesh.siMesh` object.

- `Th.find(d)` : returns the sorted indices array of the `d`-simplicial elementary meshes in the array `Th.sTh`.
- `Th.find(d,labels)` : returns the sorted indices of the `d`-simplicial elementary meshes with label in `labels`. `labels` could be an index, an array of indices. If nothing is found then return `[]`.

Several examples are given in functions:

`fc_simesh.demos.find2D()`, `fc_simesh.demos.find3D()`, `fc_simesh.demos.find3Ds()`

Now some very basic samples are presented.

Listing 6: : `fc_simesh.siMesh` find method samples

```
meshfile=fc_oogmsh.gmsh.buildmesh3d('quart_sphere2',5, 'verbose',0);
Th=fc_simesh.siMesh(meshfile, 'dim',3);
disp(Th)
idx=Th.find(3);
fprintf('3-simplices \siMeshElt_\n_\indices: \[%s], ...
labels=\[%s]\n', num2str(idx), num2str(Th.sThlab(idx)) )
idx=Th.find(2);
fprintf('2-simplices \siMeshElt_\n_\indices: \[%s], ...
labels=\[%s]\n', num2str(idx), num2str(Th.sThlab(idx)) )
idx=Th.find(2,4);
fprintf('2-simplices \siMeshElt_with_label==4\n_\indices: \[%s], ...
labels=\[%s]\n', num2str(idx), num2str(Th.sThlab(idx)) )
idx=Th.find(2,[6,4,2,10]);
fprintf('2-simplices \siMeshElt_with_label in \[6,4,2,10]\n_\indices: \[%s], ...
labels=\[%s]\n', num2str(idx), num2str(Th.sThlab(idx)) )
```

Output

```
fc_simesh.siMesh with properties:
  bbox: [ -1 1 0 1 0 1 ] (1x6 double)
  d: 3 double
  dim: 3 double
  nq: 1172 double
  nqParents: 1172 double
  nsTh: 23 double
  other: (1x1 struct)
  sTh: (1x23 cell)
  sThPhysicalTags: []
  sThboundlabs: []
  sThcolors: (23x3 double)
  sThgeolab: []
  sThlab: [ 1 2 1 2 3 4 5 6 7 1 2 3 4 5 6 7 8 9 1 2 3 4 5 ] (1x23 double)
  sThpartlabs: []
  sThphyslab: [ 1 2 1 2 3 4 5 6 7 1 2 3 4 5 6 7 8 9 1 2 3 4 5 ] (1x23 double)
  sThsimp: [ 3 3 2 2 2 2 2 2 1 1 1 1 1 1 1 1 0 0 0 0 0 ] (1x23 double)
  toGlobal: (1x1172 double)
  toParent: (1x1172 double)
  toParents: (1x1 cell)
3-simplices siMeshElt
  indices: [1 2], labels=[1 2]
2-simplices siMeshElt
  indices: [3 4 5 6 7 8 9], labels=[1 2 3 4 5 6 7]
2-simplices siMeshElt with label==4
  indices: [6], labels=[4]
2-simplices siMeshElt with label in [6,4,2,10]
  indices: [8 6 4], labels=[6 4 2]
```

### 3.4.3 feval method

Evaluates a vectorized function at vertices of the mesh. We denote by `Th` a `fc_simesh.siMesh` object.

- `res=Th.feval(fun)` : the input parameter `fun` is either a function or a cell array of function handles for vector-valued functions. If `fun` is a function then the output is an `Th.nq`-by-1 array. If `fun` is a cell array of function handles then the output is an `Th.nq`-by-`length(fun)` array.
- `res=Th.feval(fun,key,value,...)` specifies function options using one or more `key,value` pair arguments. The string `key` options could be
  - `d` : to specify the `d`-simplicial elementary meshes on which to evaluate the function (default `Th.d`). A zero value is set on all vertices not in these elementary meshes.
  - `labels` : to specify the labels of the elementary meshes on which to evaluate the function (default is all). A zero value is set on all vertices not in these elementary meshes.

Several examples are given in functions:

`fc_simesh.demos.feval2D01()`, `fc_simesh.demos.feval3D01()`, ...

We present now some very basic samples.

**Sample 1** Let  $g : \mathbb{R}^2 \mapsto \mathbb{R}$  defined by  $g(x, y) = \cos(x) \sin(y)$ . We propose in Listing 7 four approaches to defined this function for using with **feval** method.

Listing 7: : **feval** method, four ways to defined a function

```
meshfile=fc_oogmsh.gmsh.buildmesh2d('condenser11',50,'verbose',0);
Th=fc_simesh.siMesh(meshfile);

g1=@(x,y) cos(x).*sin(y); % .* for vectorized function
g2=@(X) cos(X(1,:)).*sin(X(2,:));

z1=Th.feval(g1);
z2=Th.feval(g2);

fprintf('max(abs(z2-z1))=%e\n',max(abs(z2-z1)))
```

Output

```
max(abs(z2-z1))=0.000000e+00
```

**Sample 2**

Listing 8: : **feval** method with a vector-valued function

```
meshfile=fc_oogmsh.gmsh.buildmesh2d('condenser11',50,'verbose',0);
Th=fc_simesh.siMesh(meshfile)

% f : R^2 -> R^3
f=@(x,y) cos(2*x).*sin(3*y),@(x,y) cos(3*x).*sin(4*y),@(x,y) cos(4*x).*sin(5*y);
z=Th.feval(f);
fprintf('*** nq=%d, size(z)=[%d,%d]',Th.nq,size(z))
```

Output

```
Th =

fc_simesh.siMesh with properties:
    bbox: [-1 1 -1 1] (1x4 double)
        d: 2 double
        dim: 2 double
        nq: 11945 double
    nqParents: 11945 double
        nsTh: 19 double
        other: (1x1 struct)
        sTh: (1x19 cell)
    sThPhysicalTags: []
    sThboundlabs: []
        sThcolors: (19x3 double)
        sThgeolab: []
        sThlab: [ 2 4 6 8 10 20 1 2 3 4 5 6 7 8 20 101 102 103 104 ] (1x19 double)
    sThpartlabs: []
    sThphyslab: [ 2 4 6 8 10 20 1 2 3 4 5 6 7 8 20 101 102 103 104 ] (1x19 double)
        sThsimp: [ 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 ] (1x19 double)
    toGlobal: (1x11945 double)
    toParent: (1x11945 double)
    toParents: (1x1 cell)

*** nq=11945, size(z)=[11945,3]
```

### 3.4.4 eval method

Evaluates numerical datas or vectorized functions at vertices of the mesh. We denote by `Th` a `fc_simesh.siMesh` object and  $n_q = \text{Th.nq}$  the total number of vertices.

- `res=Th.eval(data)` : the input parameter `data` could be

- a scalar,
- a handle to a vectorized function,
- a  $n_q$ -by-1 array,
- a 1-by- $m$  cell array of mixed previous kinds, ( $m \geq 1$ ).

The return value is a  $n_q$ -by-1 array if the input parameter `data` is not a cell array otherwise it's a  $n_q$ -by- $m$  array.

- `res=Th.eval(data,key,value,...)` specifies function options using one or more key,value pair arguments. The string `key` options could be
  - `d` : to specify the  $d$ -simplicial elementary meshes on which to evaluate `data` (default `Th.d`). A zero value is set on all vertices not in these elementary meshes.
  - `labels` : to specify the labels of the elementary meshes on which to evaluate `data` (default is all). A zero value is set on all vertices not in these elementary meshes.

Several examples are given in functions:

`fc_simesh.demos.eval2D01()`, `siMesh.demos.eval3D01()`, ...

We present now some very basic samples.

### Sample 1

Listing 9: : eval method, four ways to defined a function

```
meshfile=fc_oogmsh.gmsh.buildmesh2d('condenser11',50,'verbose',0);
Th=fc_simesh.siMesh(meshfile);

g1=pi*ones(Th.nq,1);
g2=pi*ones(1,Th.nq);
g3=@(X) pi;

z1=Th.eval(g1);
z2=Th.eval(g2);
z3=Th.eval(g3);

fprintf(' size (z1)=[%d,%d]\n',size(z1))
fprintf(' size (z2)=[%d,%d]\n',size(z2))
fprintf(' size (z3)=[%d,%d]\n',size(z3))
fprintf(' max( abs (z2-z1) )=%e\n',max(abs(z2-z1)))
fprintf(' max( abs (z3-z1) )=%e\n',max(abs(z3-z1)))
```

Output

```
size(z1)=[11945,1]
size(z2)=[11945,1]
size(z3)=[11945,1]
max(abs(z2-z1))=0.000000e+00
max(abs(z3-z1))=0.000000e+00
```

### Sample 2

Listing 10: : eval method with a vector-valued function

```
meshfile=fc_oogmsh.gmsh.buildmesh2d('condenser11',50,'verbose',0);
Th=fc_simesh.siMesh(meshfile);
u=Th.feval(@(x,y) cos(3*x).*sin(4*y));
% f : R^2 -> R^3
f=@(x,y) cos(2*x).*sin(3*y),u,@(x,y) cos(4*x).*sin(5*y),pi;
z=Th.eval(f);
fprintf(' ***_nq=%d, _size (z)=[%d,%d]',Th.nq,size(z))
```

Output

```
*** nq=11945, size(z)=[11945,4]
```

### 3.4.5 get\_h method

returns the maximum edges length of the mesh. We denote by `Th` a `fc_simesh.siMesh` object.

- `h=Th.get_h()`

### 3.4.6 get\_mesh method

Returns a vertices array `q`, a connectivity array `me` and a `toGlobal` indices array.

- `[q,me,toGlobal]=Th.get_mesh()` : returns the global vertices array `q`, the connectivity array `me` (i.e. all the `Th.d`-simplices of the mesh). In this case, `toGlobal` is just `1:Th.nq`.
- `[q,me,toGlobal]=Th.get_mesh(key,value,...)` specifies function options using one or more `key,value` pair arguments. The string `key` options could be
  - `'d'` : to specify the `d`-simplicial elementary meshes to consider.
  - `'labels'` : to specify the labels of the elementary meshes to consider.

In this case, `toGlobal` is a 1-by-`length(q)` array (subset of `1:Th.nq`). If we denote by `qglob` the global vertices array then  
`qglob(:,toGlobal)==q`

Several examples are given in functions:

`fc_simesh.demos.get_mesh2D()`, `siMesh.demos.get_mesh3D()`, `siMesh.demos.get_mesh3Ds()`

Listing 11: : `get_mesh` method, four ways to defined a function

```
meshfile=fc_oogmsh.gmsh.buildmesh2d('condenser11',50,'verbose',0);
Th=fc_simesh.siMesh(meshfile);

[q,me,toGlobal]=Th.get_mesh();
[q2,me2,toGlobal2]=Th.get_mesh('labels',2:2:8);
[q1,me1,toGlobal1]=Th.get_mesh('d',1,'labels',1:8);

fprintf('norm(q(:,toGlobal2)-q2,Inf)=%e\n',norm(q(:,toGlobal2)-q2,Inf))
fprintf('norm(q(:,toGlobal1)-q1,Inf)=%e\n',norm(q(:,toGlobal1)-q1,Inf))
```

Output

```
norm(q(:,toGlobal2)-q2,Inf)=0.000000e+00
norm(q(:,toGlobal1)-q1,Inf)=0.000000e+00
```

### 3.4.7 get\_nme method

Returns the number of  $d$ -simplicial elements with  $d = \mathcal{T}_h.d$  by default. We denote by `Th` a `fc_simesh.siMesh` object.

- `nme=Th.get_nme()` : returns the number of `Th.d`-simplicial elements in the mesh.
- `nme=Th.get_mesh(key,value,...)` specifies function options using one or more `key,value` pair arguments. The string `key` options could be
  - `'d'` : to specify the `d`-simplicial elementary meshes to consider.

- 'labels' : to specify the labels of the elementary meshes to consider.

Listing 12: : get\_nme method

```
meshfile=fc_oogmsh.gmsh.buildmesh3d('quart_sphere2',5);
Th=fc_simesh.siMesh(meshfile);
for d=[Th.d;-1:0]
    fprintf('Number_of_%d-simplices_in_%d\n',d,Th.get_nme('d',d))
end

nme=Th.get_nme('d',2,'labels',1:4);
fprintf('Number_of_2-simplices_in_union_of_label''s_1_to_4_:_%d\n',nme);
```

Output

```
[fc-oogmsh] Input file : <fc-oogmsh>/geodir/3d/quart_sphere2.geo
[fc-oogmsh] Mesh file <fc-oogmsh>/meshes/quart_sphere2-5.msh [version 4.1] already exists.
-> Use "force" flag to rebuild if needed.
Mesh quart_sphere2-5.msh is a 3-dimensional mesh
Force dimension to 3
Number of 3-simplices : 4778
Number of 2-simplices : 1653
Number of 1-simplices : 115
Number of 0-simplices : 5
Number of 2-simplices in union of label's 1 to 4 : 748
```

### 3.4.8 get\_nq method

Returns the number of vertices in the union of some elementary meshes. By default all the (Th.d)-simplicial elementary meshes are selected. We denote by Th a `fc_simesh.siMesh` object.

- `nq=Th.get_nq()` : returns the number of vertices in the union of the Th.d-simplicial elementary meshes.
- `nq=Th.get_nq(key,value,...)` specifies function options using one or more key,value pair arguments. The string key options could be
  - 'd' : to specify the d-simplicial elementary meshes to consider.
  - 'labels' : to specify the labels of the elementary meshes to consider.

Listing 13: : get\_nqe method

```
meshfile=fc_oogmsh.gmsh.buildmesh3d('quart_sphere2',5);
Th=fc_simesh.siMesh(meshfile);
for d=[Th.d;-1:0]
    fprintf('Number_of_vertices_in_%d-simplices_elementary_meshes_:_%d\n',d,Th.get_nq('d',d))
end

nq=Th.get_nq('d',2,'labels',1:4);
fprintf('Number_of_vertices_in_the_union_of_2-simplices_elementary_meshes_of_label''s_1_to_4_:_%d\n',nq);
```

Output

```
[fc-oogmsh] Input file : <fc-oogmsh>/geodir/3d/quart_sphere2.geo
[fc-oogmsh] Mesh file <fc-oogmsh>/meshes/quart_sphere2-5.msh [version 4.1] already exists.
-> Use "force" flag to rebuild if needed.
Mesh quart_sphere2-5.msh is a 3-dimensional mesh
Force dimension to 3
Number of vertices in 3-simplices elementary meshes : 1172
Number of vertices in 2-simplices elementary meshes : 812
Number of vertices in 1-simplices elementary meshes : 111
Number of vertices in 0-simplices elementary meshes : 5
Number of vertices in the union of 2-simplices elementary meshes of label's 1 to 4 : 405
```

### 3.4.9 get\_labels method

Returns the labels of the  $d$ -simplicial elementary meshes. We denote by  $Th$  a `fc_simesh.siMesh` object.

- `labels=Th.get_labels(d)` : the labels of the  $d$ -simplicial elementary meshes.

Listing 14: : get\_labels method

```
geofile=fc_simesh.get_geo(3,3,'quart_sphere2');
meshfile=fc_oogmsh.gmsh.buildmesh3d(geofile,10);
Th=fc_simesh.siMesh(meshfile);
lab3=Th.get_labels(3)
lab2=Th.get_labels(2)
lab2=Th.get_labels(1)
lab2=Th.get_labels(0)
```

Output

```
[fc-oogmsh] Input file : <fc-oogmsh>/geodir/3d/quart_sphere2.geo
[fc-oogmsh] Starting building mesh <fc-oogmsh>/meshes/quart_sphere2-10.msh with gmsh 4.11.0
[fc-oogmsh] Using command : gmsh -3 -setnumber N 10 -string "Mesh.MshFileVersion=4.1;" ...
<fc-oogmsh>/geodir/3d/quart_sphere2.geo -o <fc-oogmsh>/meshes/quart_sphere2-10.msh
Be patient...
[fc-oogmsh] Using gmsh 4.11.0 to write MSH file format version 4.1 in ...
<fc-oogmsh>/meshes/quart_sphere2-10.msh
Mesh quart_sphere2-10.msh is a 3-dimensional mesh
Force dimension to 3
lab3 =

1 2

lab2 =

1 2 3 4 5 6 7

lab2 =

1 2 3 4 5 6 7 8 9

lab2 =

1 2 3 4 5
```

We can refer to Figure 2 to validate the output results.

### 3.4.10 move method

Moving a mesh. We denote by  $Th$  a `fc_simesh.siMesh` object.

- `Th.move(u,dims)` : the input parameter  $u$  is the displacement vector which is either a numerical array or a cell array of numerical array. The second parameter  $dims$  is uses to specify the dimensions where displacement vector is applied.
  - Let  $U=u$  if  $u$  is an  $n$ -by- $n_q$  array and  $U=u.'$  if  $u$  is an  $n_q$ -by- $n$  array. Then  $dims$  is a vector of length  $n$  and all nodes array  $q$  in  $Th$  are replaced by

$$Th.sThi.q(dims,:)=q(dims,:)+U(:,Th.sThi.toGlobal))$$

- $u$  is a cell array of length  $d$ . Let  $U=u\{k\}$  if  $u\{k\}$  is an 1-by- $n_q$  array and  $U=u\{k\}.'$  if  $fcmcodeu\{i\}$  is an  $n_q$ -by-1 array.

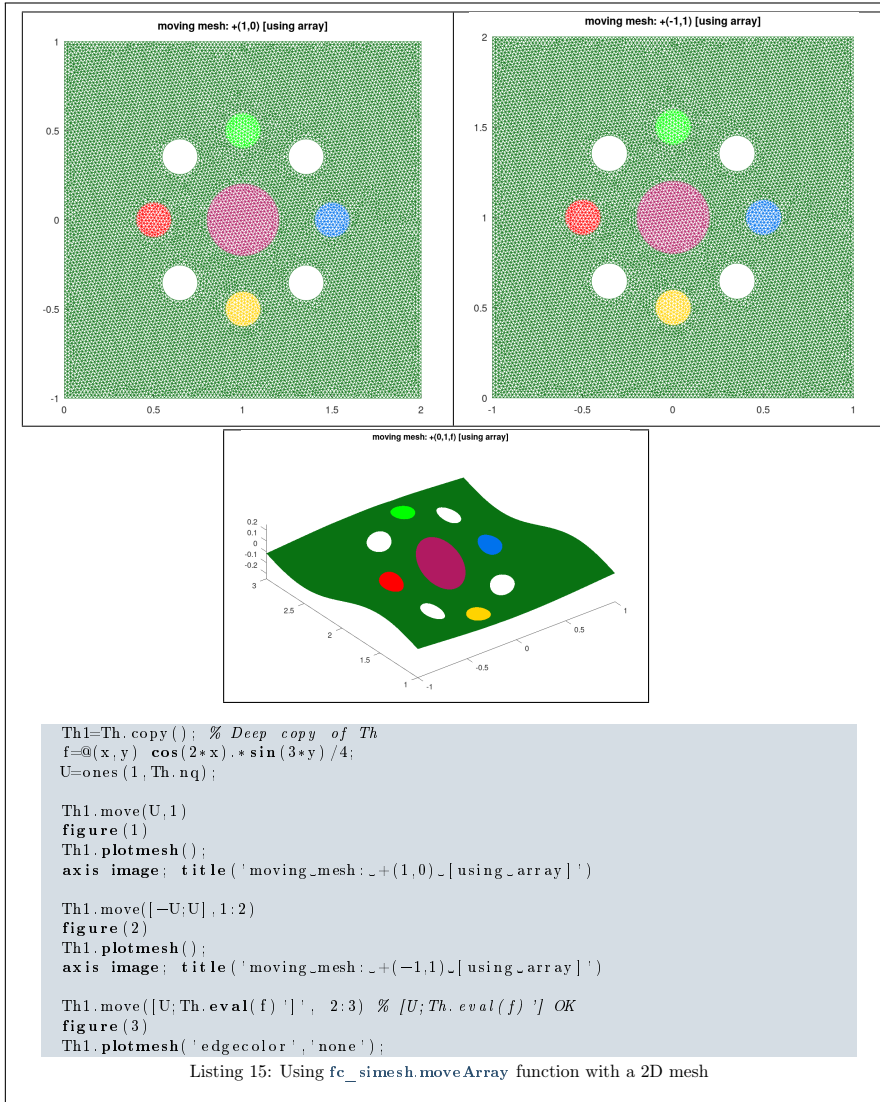
$$Th.sThi.q(dims(k),:)=q(dims(k),:)+U(:,Th.sThi.toGlobal))$$



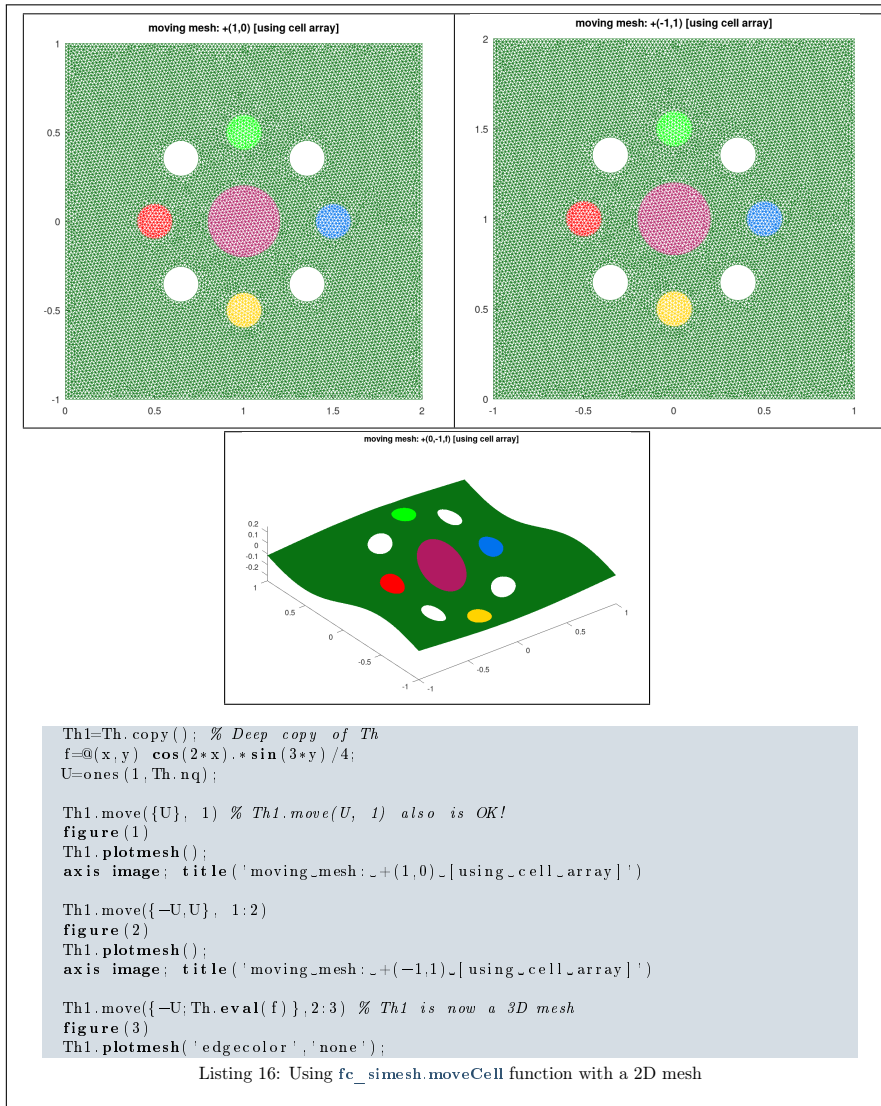
**remark 3.1**

1. Take care that modification in `Th` are done *inplace*. One can use the command `Th1=Th.copy()` which make a deep copy of the `Th` object (modifying one object not change its deep copy).
2. If `max(dims)` greather than `Th.dim`, then the mesh dimension is automatically increased and new dimensions in nodes array are set to zeros.

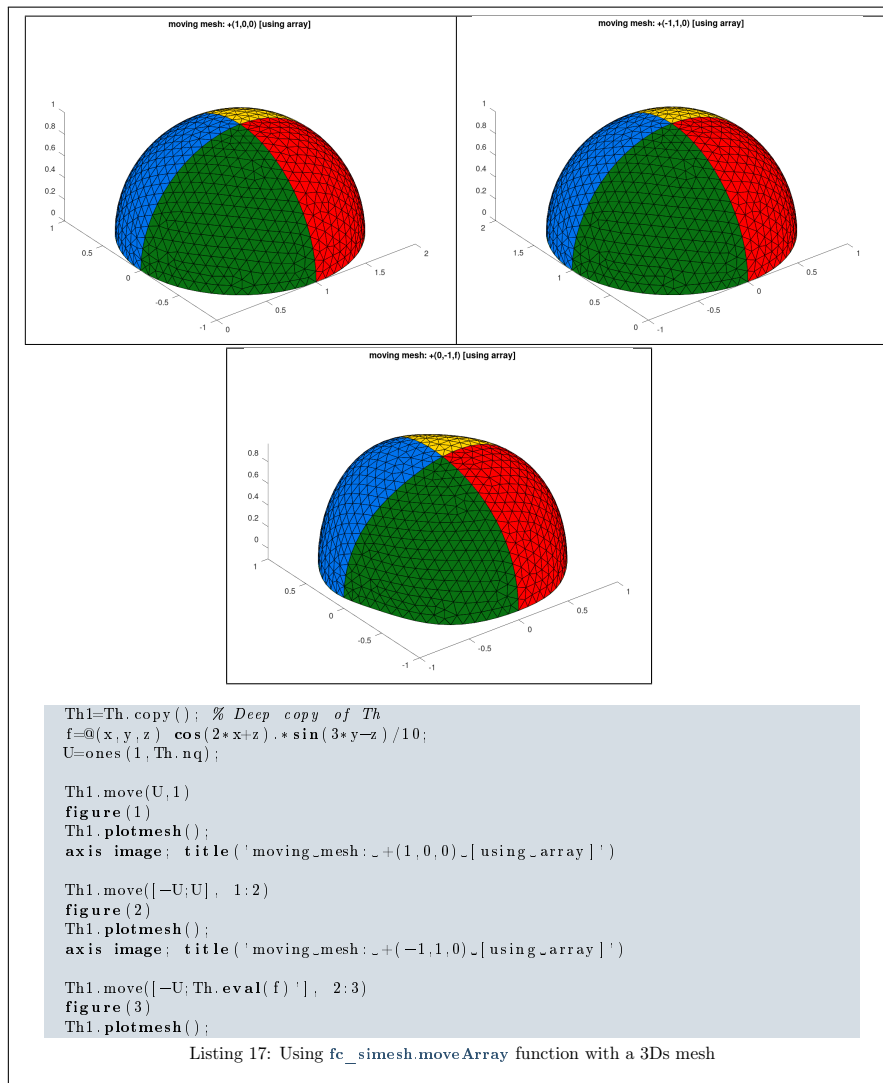
**2D example :** the following code is part of the `fc_simesh.demos.moveArray2D` function.



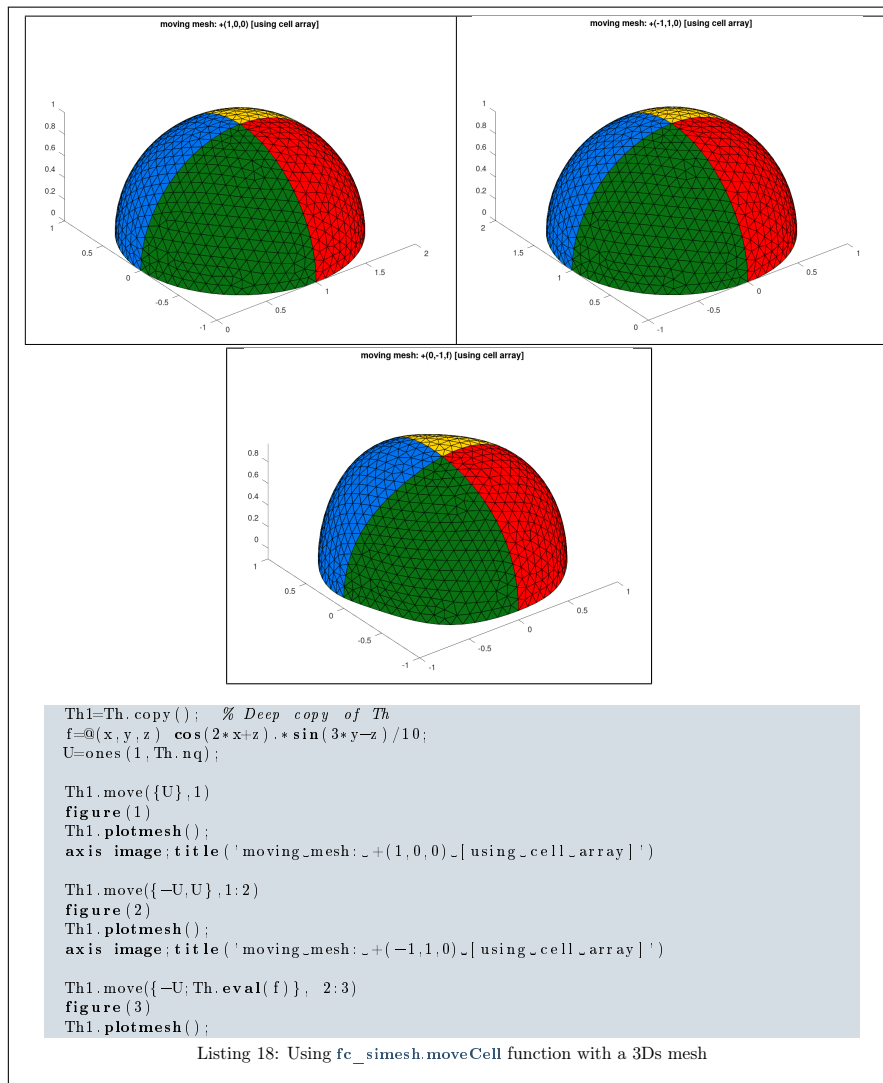
**2D example :** the following code is part of the `fc_simesh.demos.moveCell2D` function.



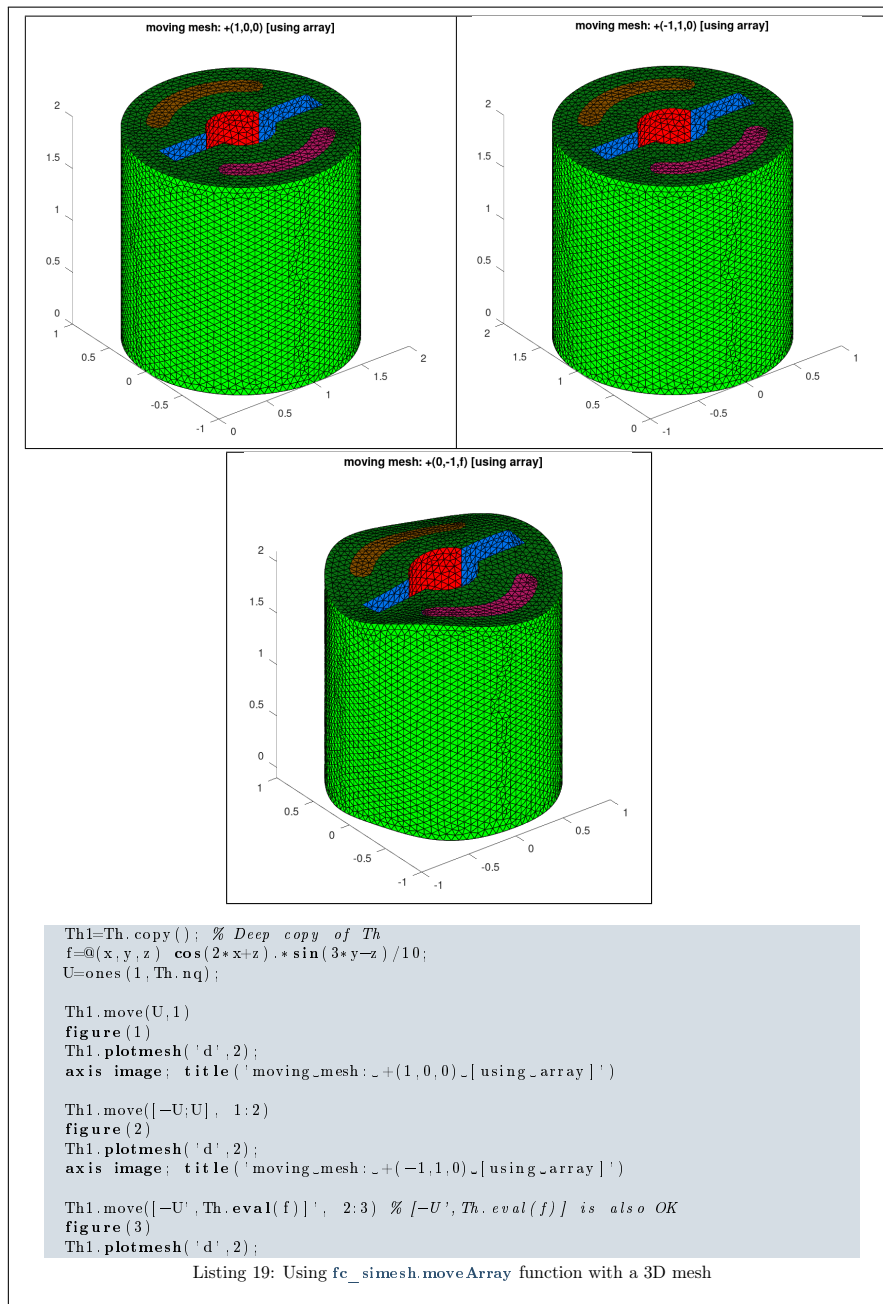
**3Ds example :** the following code is part of the `fc_simesh.demos.moveArray3Ds` function.



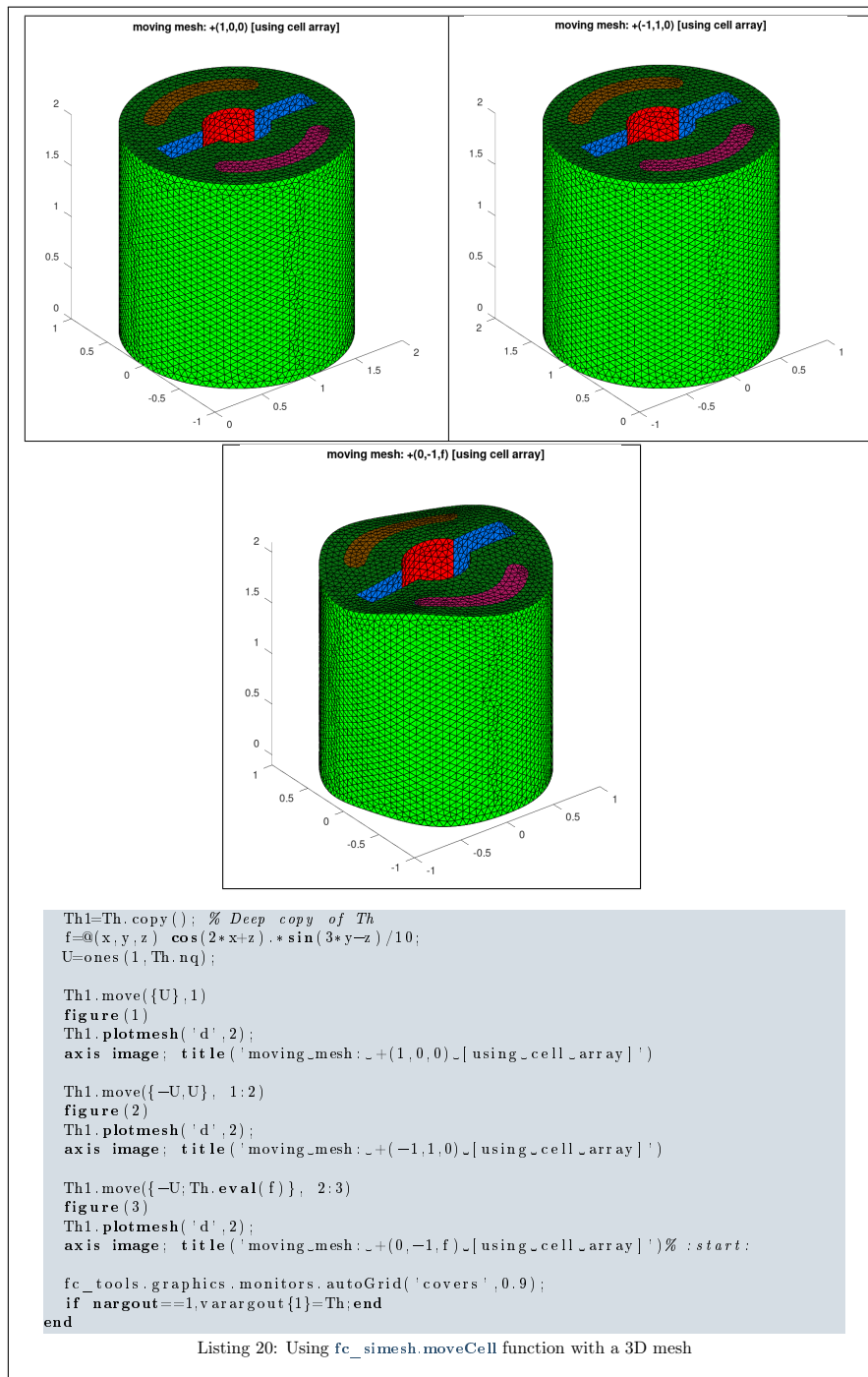
**3Ds example :** the following code is part of the `fc_simesh.demos.moveCell3Ds` function.



**3D example :** the following code is part of the `fc_simesh.demos.moveArray3D` function.



**3D example :** the following code is part of the `fc_simesh.demos.moveCell3D` function.



### 3.4.11 plotmesh method

The `plotmesh` method displays the mesh or parts of the mesh defined by an `fc_simesh.siMesh` object.

## Syntaxe

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

## Description

`Th.plotmesh()` displays all the (Th.d)-dimensional simplices elements of Th, a `fc_simesh.siMesh` object.

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

- 'd' : to specify the dimension of the simplices elements (default : Th.d)
- 'labels' : to select the labels of the elements to display,
- 'color' : to specify the color of the displayed mesh elements. (default : use one color by displayed mesh elements),
- 'inlegend' : add a legend name to graph if true (default : **false**)
- 'bounds' : If **true**, draw the borders of the selected elementaries mesh elements (only for 2-dimensional simplices). (default : **false**)
- 'cutPlane' : cut mesh by  $n$  plans 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. (only for simplices in dimension 3) default : `[]` (no cut).

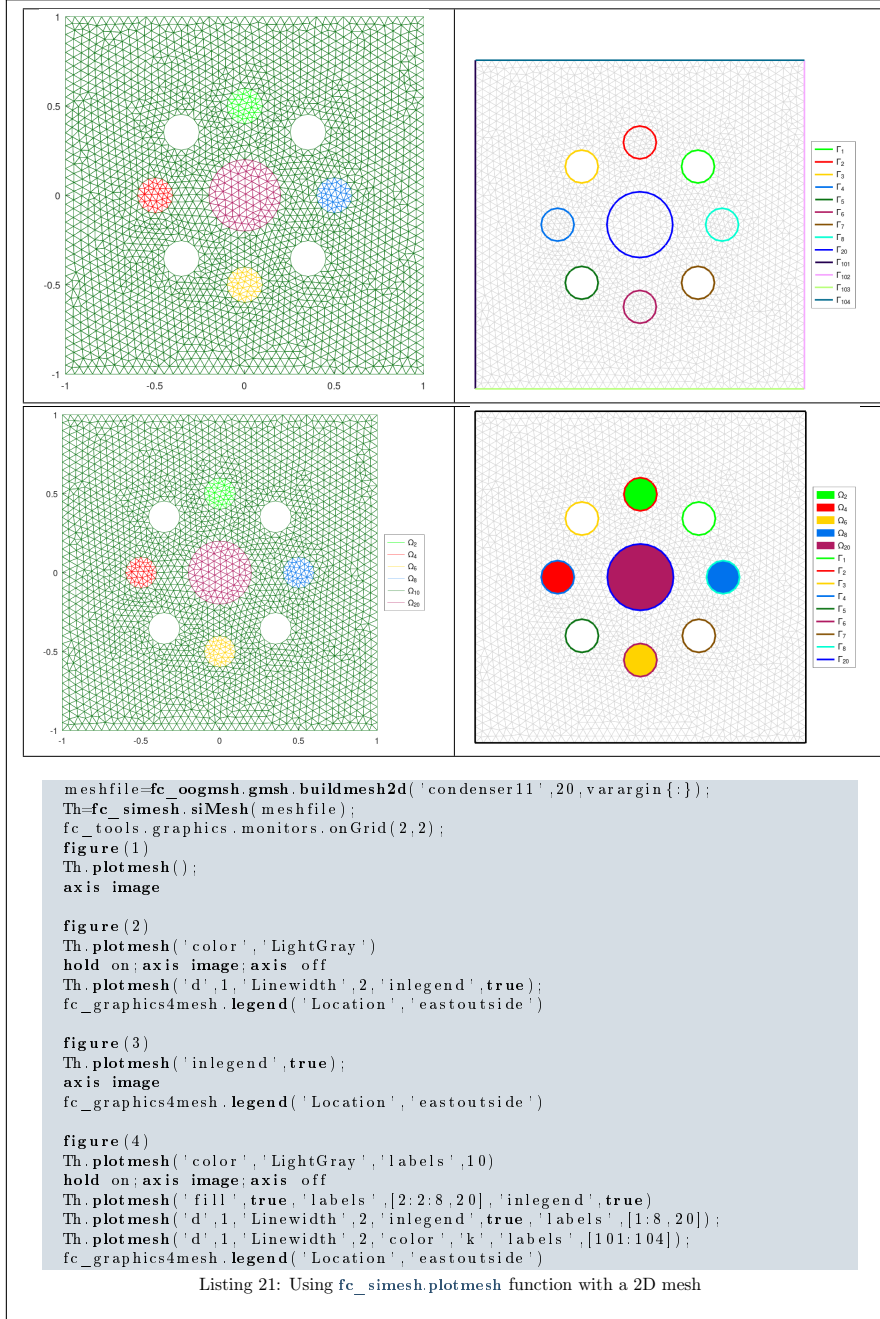
The options of second level depend on the type of elementaries mesh elements to represent.

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

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

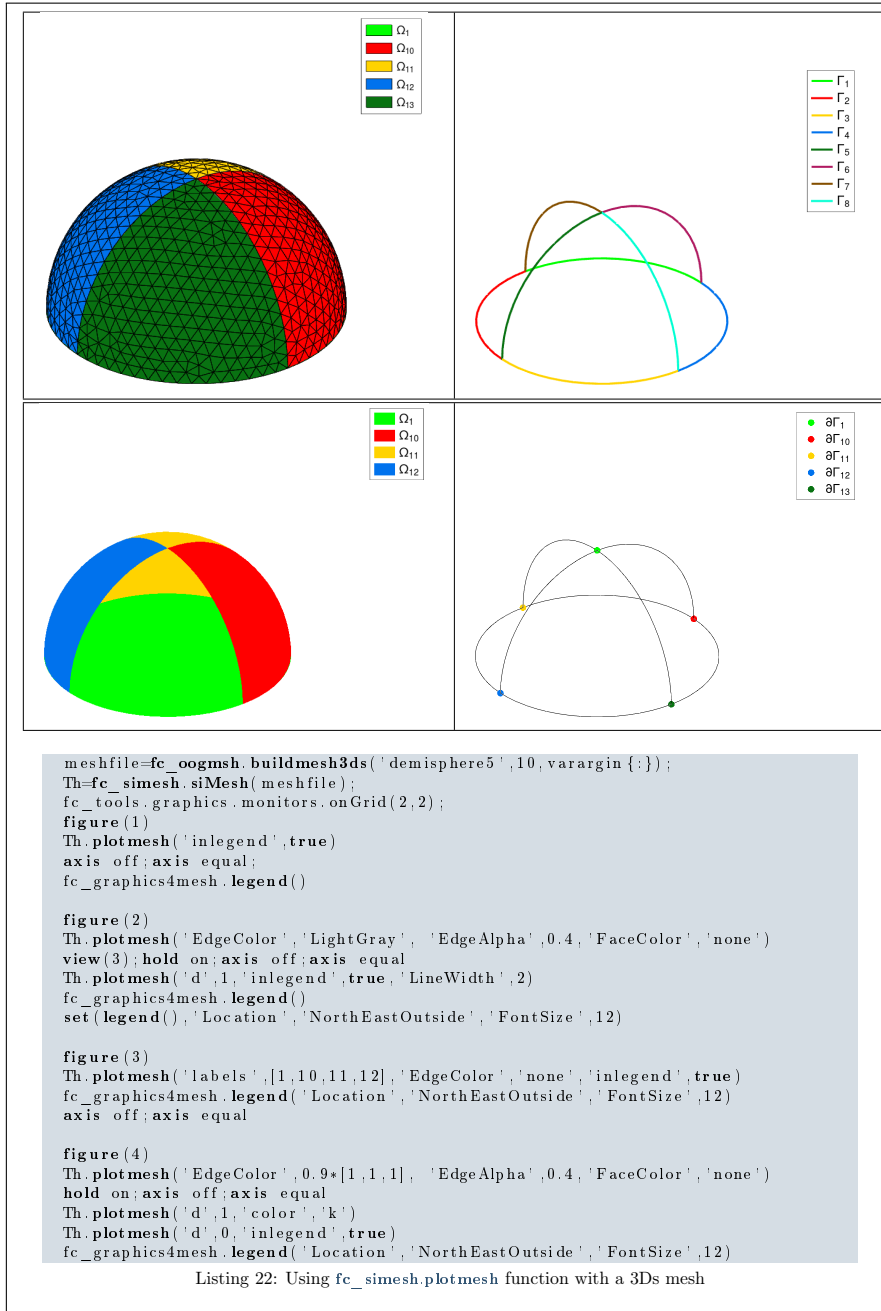


**2D example :** the following code is part of the `fc_simesh.demos.plotmesh2D` function.

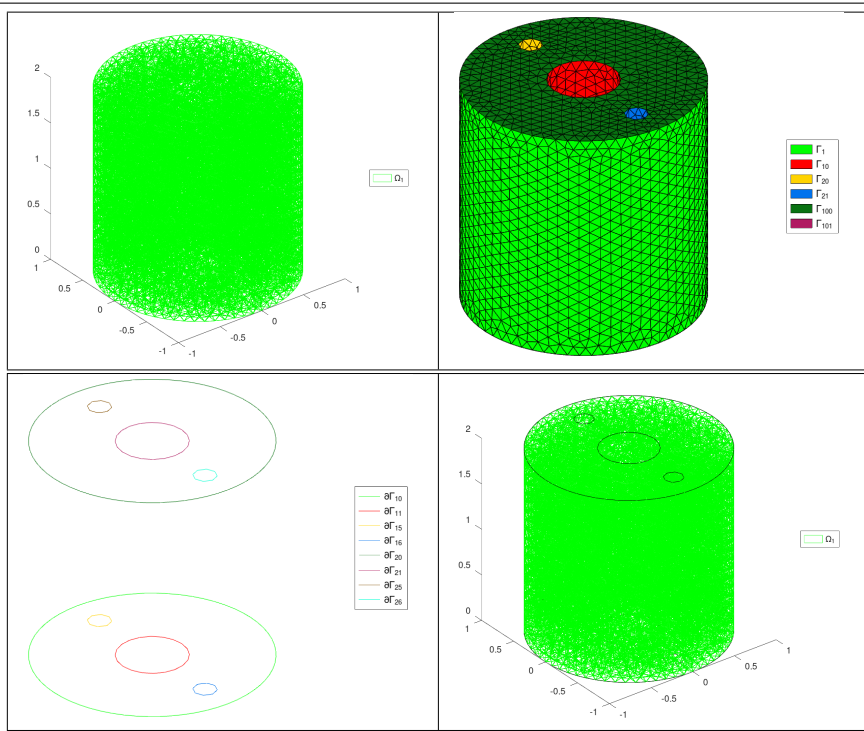


**3Ds example :** the following code is part of the `fc_simesh.demos.plotmesh3Ds` function.





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



```

meshfile=fc_oogmsh.gmsh.buildmesh3d('cylinder3holes',10,varargin{:});
Th=fc_simesh.siMesh(meshfile);
fc_tools.graphics.monitors.onGrid(3,3,'figures',1:7);
figure(1)
Th.plotmesh('inlegend',true)
axis image
fc_graphics4mesh.legend('Location','EastOutside')

figure(2)
Th.plotmesh('d',2,'inlegend',true);
axis image;axis off
fc_graphics4mesh.legend('Location','EastOutside')

figure(3)
Th.plotmesh('d',1,'inlegend',true);
axis image;axis off
fc_graphics4mesh.legend('Location','EastOutside')

figure(4)
Th.plotmesh('inlegend',true)
hold on
Th.plotmesh('d',1,'color','k');
axis image
fc_graphics4mesh.legend('Location','EastOutside')

figure(5)
Th.plotmesh('d',2,'inlegend',true);
axis image
fc_graphics4mesh.legend('Location','EastOutside')

figure(6)
Th.plotmesh('d',2,'edgecolor',0.8*[1 1 1],'facecolor','None','edgealpha',0.5)
hold on;axis image
Th.plotmesh('d',1,'inlegend',true);
fc_graphics4mesh.legend('Location','EastOutside')

figure(7)
P=[fc_tools.graphics.PlaneCoefs([0 0 1],[0 0 1]); ...
    fc_tools.graphics.PlaneCoefs([0 0 1],[-1 0 0])];
Th.plotmesh('cutPlane',P,'Color','DarkGrey')
hold on;axis image
Th.plotmesh('d',2,'cutPlane',P,'inlegend',true);
fc_graphics4mesh.legend('Location','EastOutside')

```

Listing 23: Using `fc_simesh.plotmesh` function with a 3D mesh

### 3.4.12 plot method

The `plot` method displays scalar datas on the mesh or parts of the mesh defined by an `fc_simesh.siMesh` object. We denote by `Th` a `fc_simesh.siMesh` object.

#### Syntaxe

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

#### Description

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

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

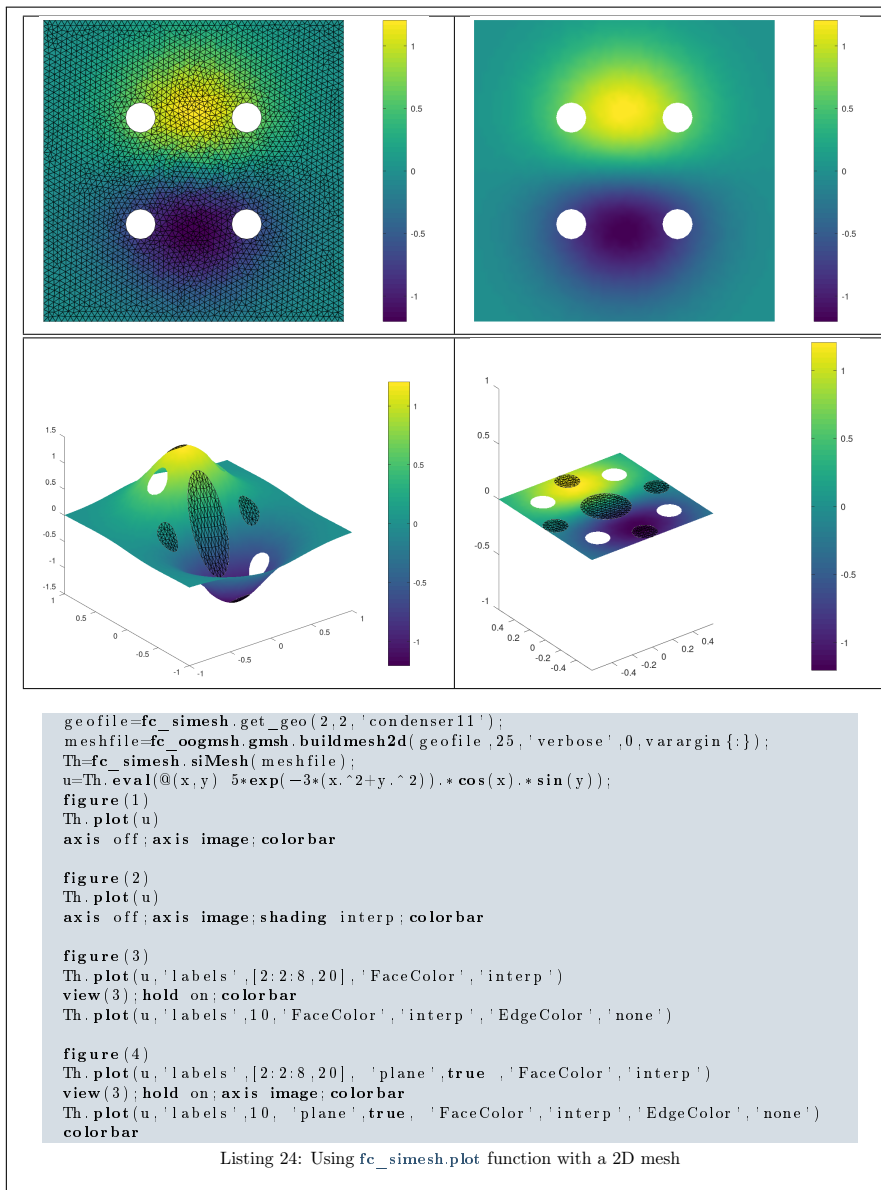
- `'d'` : to specify the dimension of the simplices elements (default : `Th.d`)
- `'labels'` : to select the labels of the elements to display data,
- `'plane'` : if true, made a 2D representation in the  $xy$ -plane, otherwise made a 3D representation with  $z$ -value set to `u` (default : `false`)

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

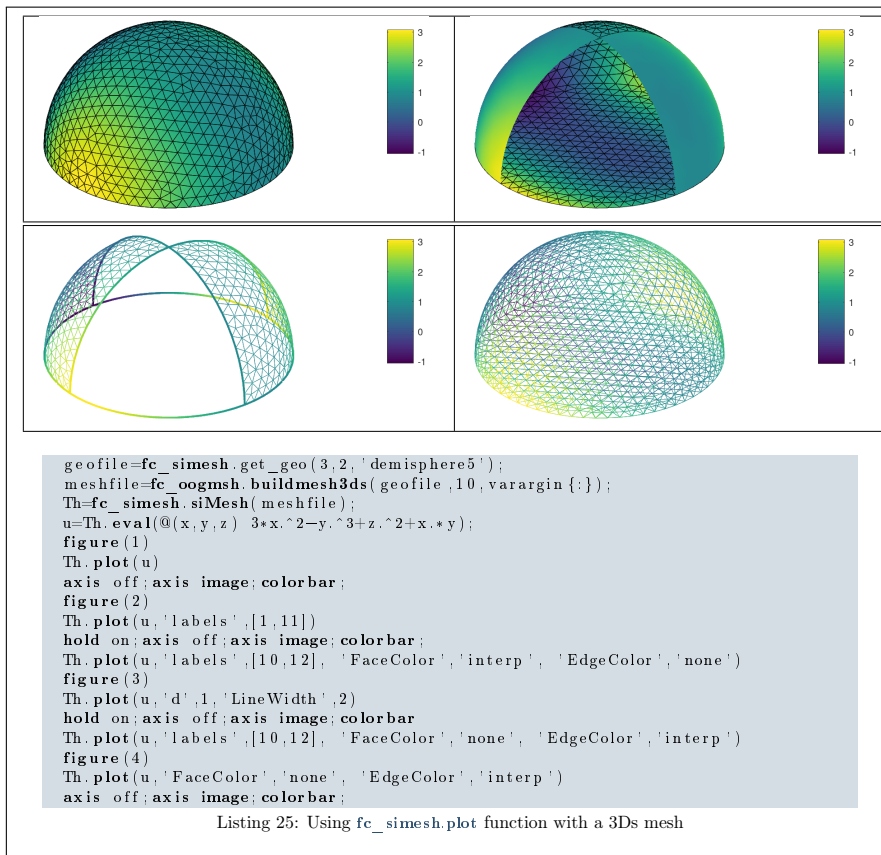
One can use any option of the following functions according to the type of  $d$ -simplex.

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

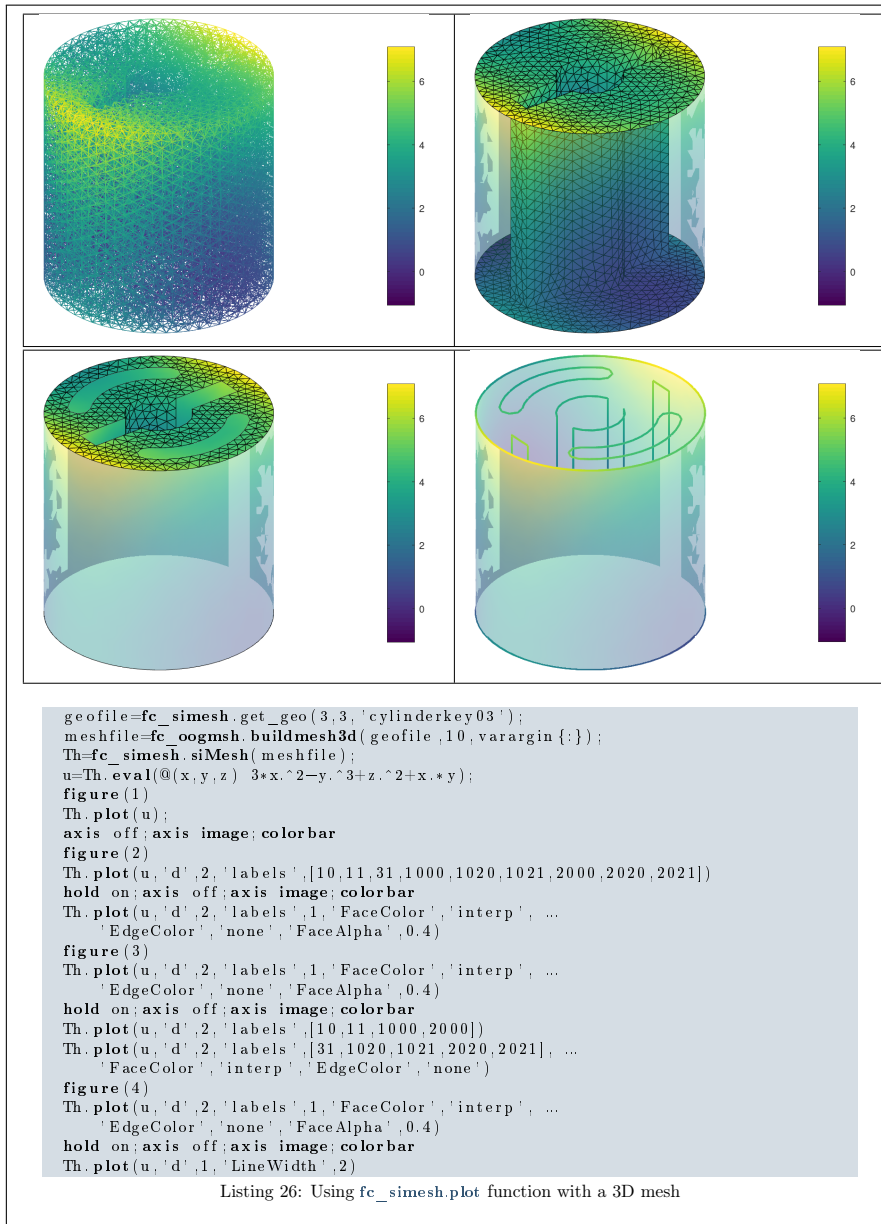
**2D example :** the following code is part of the `fc_simesh.demos.plot2D` function.



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



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



### 3.4.13 plotiso method

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

#### Syntaxe

```

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

```

## Description

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

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

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

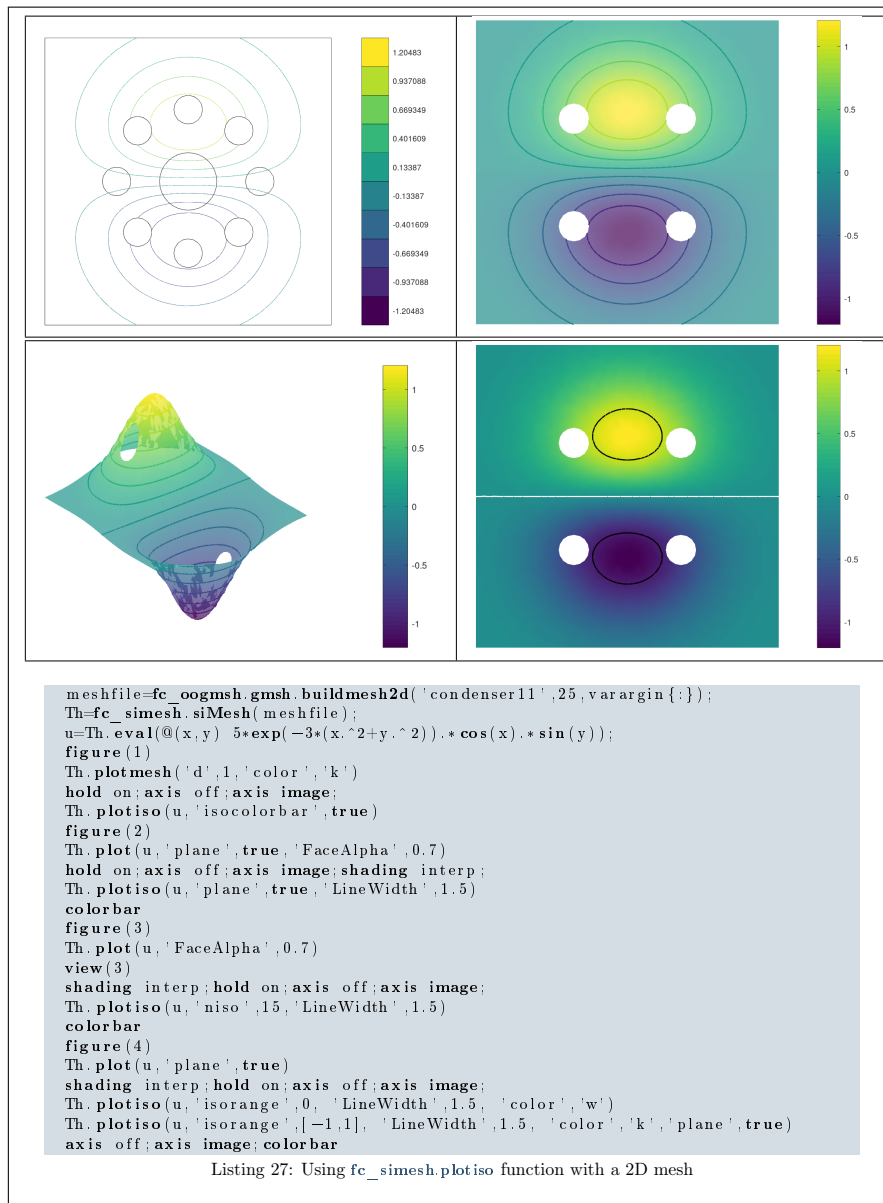
The options of second level are all options of

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

This function accepts until 4 output arguments :

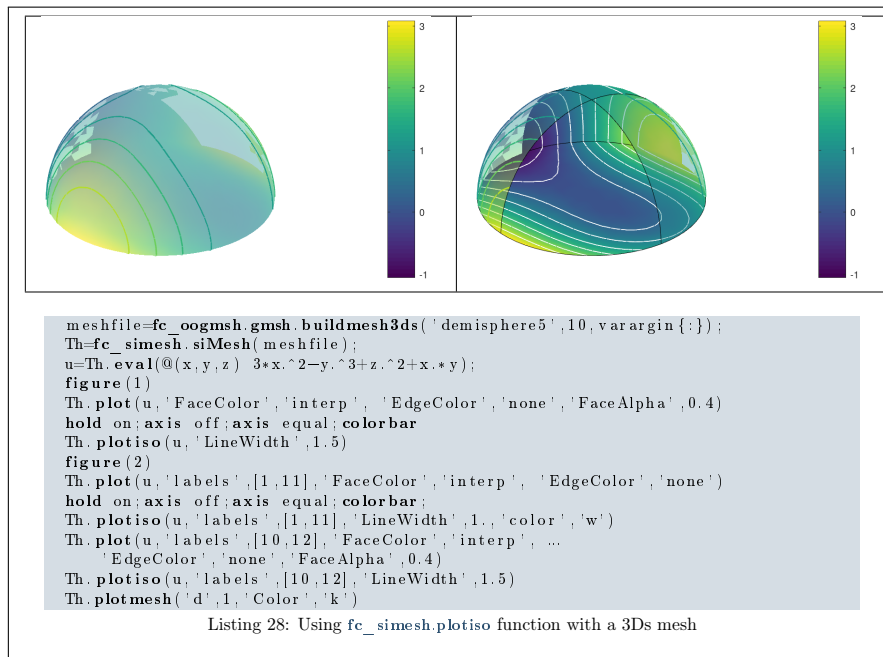
- 1st output is the colors of the isolines
- 2nd output is the isovalues of the isolines
- 3th output is the handle of the colobar iso.
- 4th output is all the handles of the isolines as an 2D-array of dimension N-by-niso, where N is the number of 2-simplex elementary meshes where isolines are drawn.

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

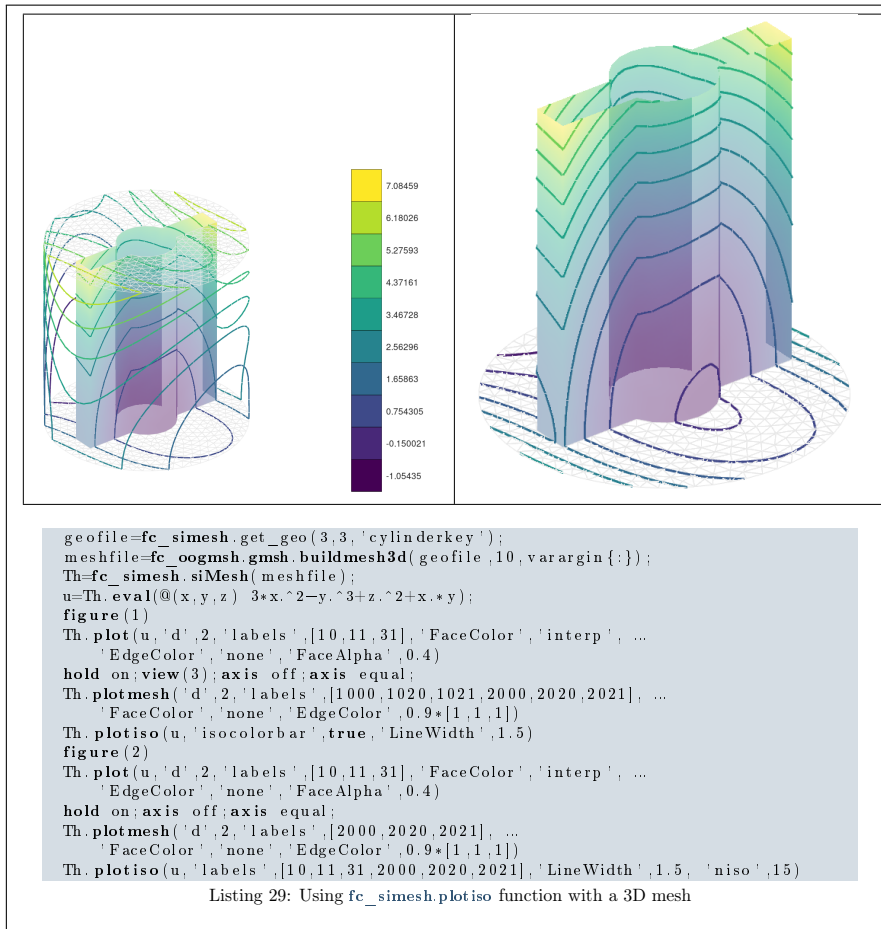


**3Ds example :** the following code is part of the `fc_simesh.demos.plotiso3Ds` function.





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



### 3.4.14 slicemesh method

The `slicemesh` method displays intersection of a plane and a 3D mesh or parts of a 3D mesh defined by an `fc_simesh.siMesh` object.

#### Syntaxe

```

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

```

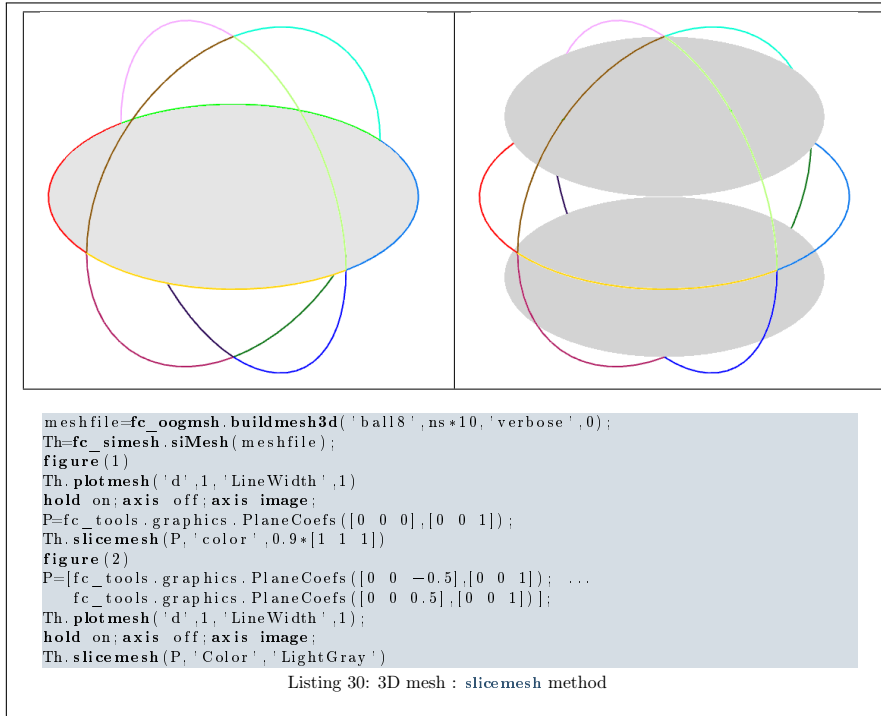
#### Description

`Th.slicemesh(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 of `Th`, a `fc_simesh.siMesh` object. To compute  $P$  one can use the function `fc_tools.graphics.PlaneCoefs` of the package. With this function, the array `P`, is obtained with `P=fc_tools.graphics.PlaneCoefs(Q,V)` where `Q` is a point in the plane and `V` is a vector orthogonal to it.

`Th.slicemesh(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 : 'lightgrey', `rgb=[0.9,0.9,0.9]`)
- 'labels' : to select the labels of the elements to intersect,

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



### 3.4.15 slice method

The method `slice` method displays datas on the intersection of a plane and a 3D mesh or parts of a 3D mesh defined by an `fc_simesh.siMesh` object.

#### Syntaxe

```

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

```

#### Description

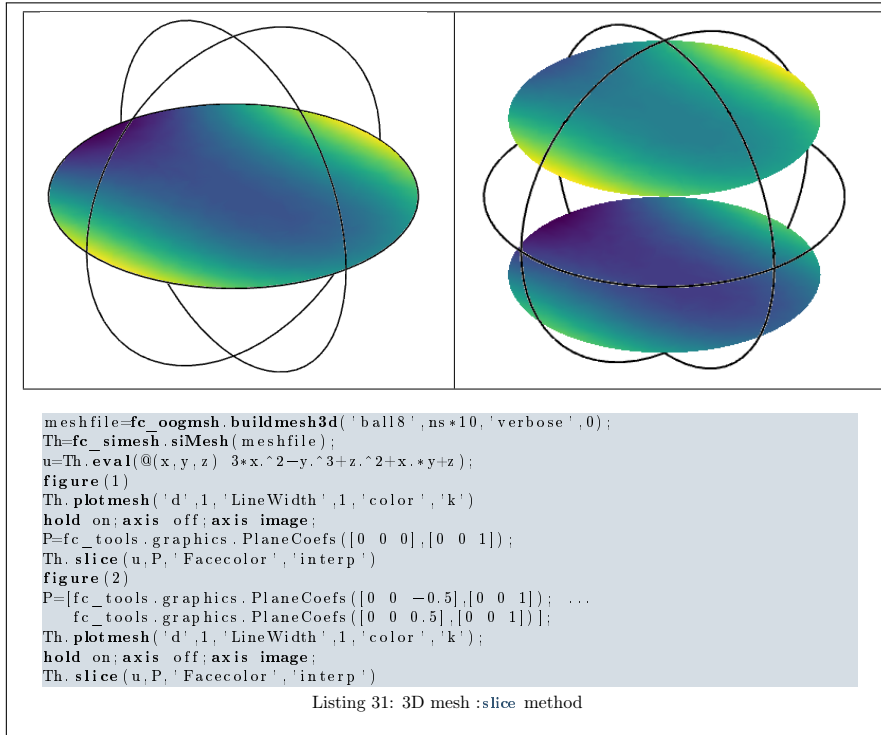
`Th.slice(u,P)` displays `u` data on the intersection of the plane defined by  $P(1)x + P(2)y + P(3)z + P(4) = 0$  and all the 3-dimensional simplices elements of `Th`, a `fc_simesh.siMesh` object. The data `u` is an 1D-array of

size `Th.nq` or `Th.nqGlobal` or `Th.nqParent`. To compute  $P$  one can use the function `fc_tools.graphics.PlaneCoefs` of the package. With this function, 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.

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

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

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



### 3.4.16 sliceiso method

The `sliceiso` method displays isolines of data on the intersection of a plane and a 3D mesh or parts of a 3D mesh defined by an `fc_simesh.siMesh` object.

#### Syntaxe

```

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

```

## Description

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

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

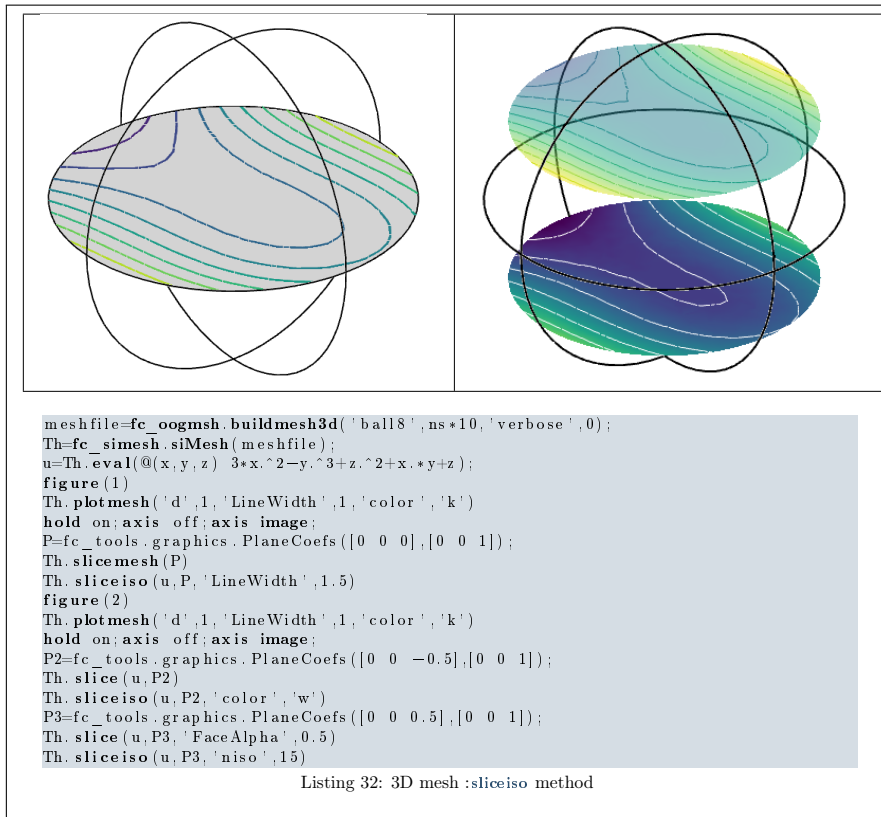
- `'labels'` : to select the labels of the elements to intersect,
- `'niso'` : to specify the number of isolines (default : 10)
- `'isorange'` : to specify the list of isovalues (default : empty)
- `'color'` : to specify one color for all isolines (default : empty)
- `'isocolorbar'` : if true display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

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

This function accepts until 4 output arguments :

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

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



### 3.4.17 plotquiver method

The **plotquiver** method displays vector field datas on the mesh or parts of the mesh defined by an **fc\_simesh.siMesh** object.

#### Syntaxe

```

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

```

#### Description

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

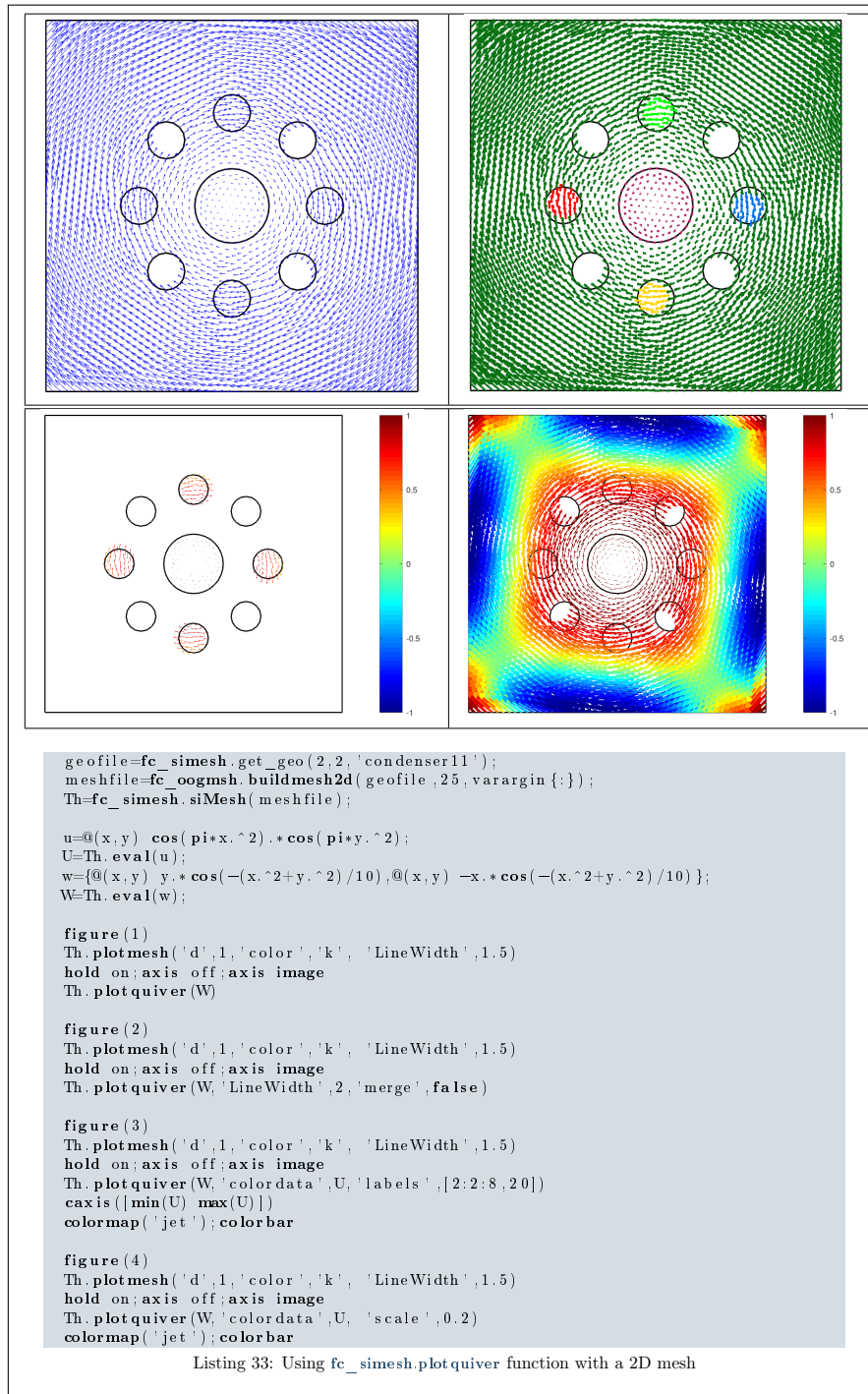
**Th.plotquiver(V,Key,Value,...)** specifies function options using one or more **Key,Value** pair arguments. Options of first level are

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

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

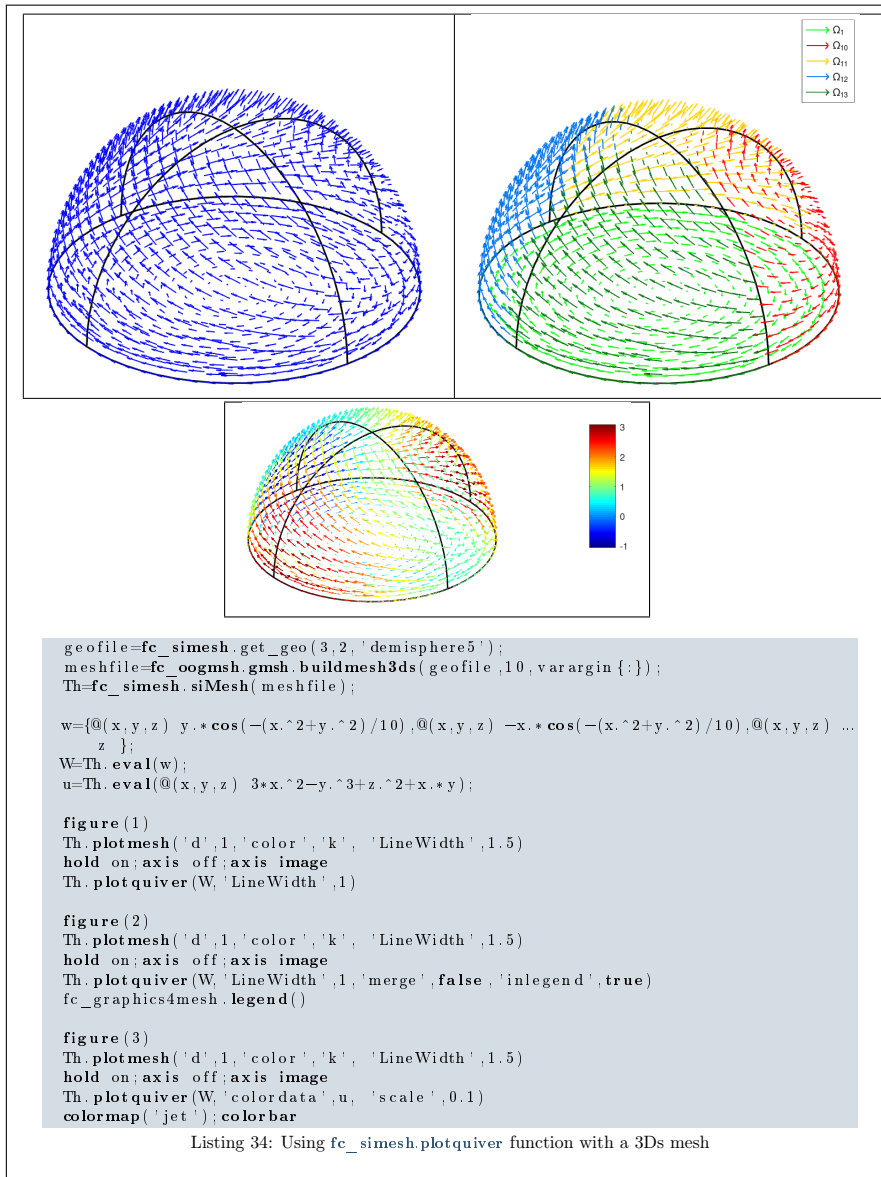
- `quiver` function in dimension 2 with an empty 'colordata'
- `quiver3` function in dimension 3 with an empty 'colordata'
- `vfield3` function in dimension 2 or 3 with 'colordata' set to an 1D-array of length `Th.nq`.

**2D example :** the following code is part of the `fc_simesh.demos.plotquiver2D` function.

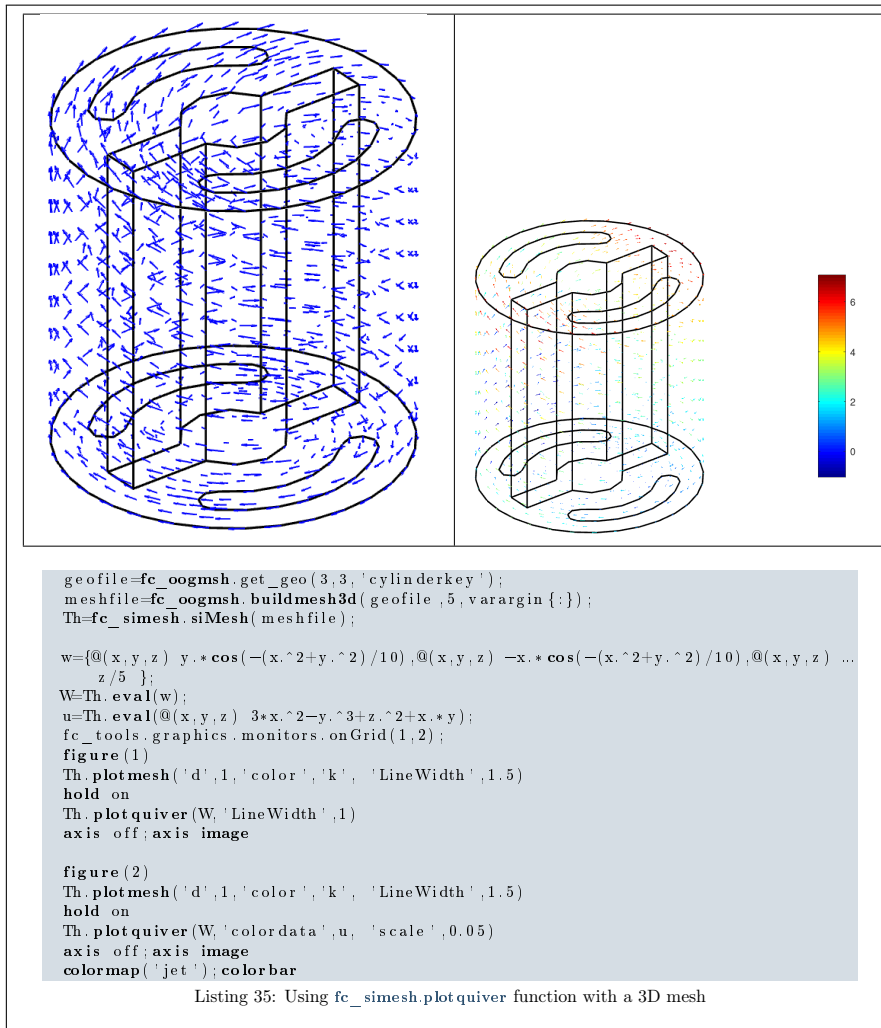


**3Ds example :** the following code is part of the `fc_simesh.demos.plotquiver3Ds` function.





**3D example :** the following code is part of the `fc_simesh.demos.plotquiver3D` function.



### 3.4.18 scatter method

The `scatter` method displays scalar datas as colored points on the mesh or parts of the mesh defined by an `fc_simesh.siMesh` object.

#### Syntaxe

```

Th.scatter(u)
Th.scatter(u,Name,Value,...)

```

#### Description

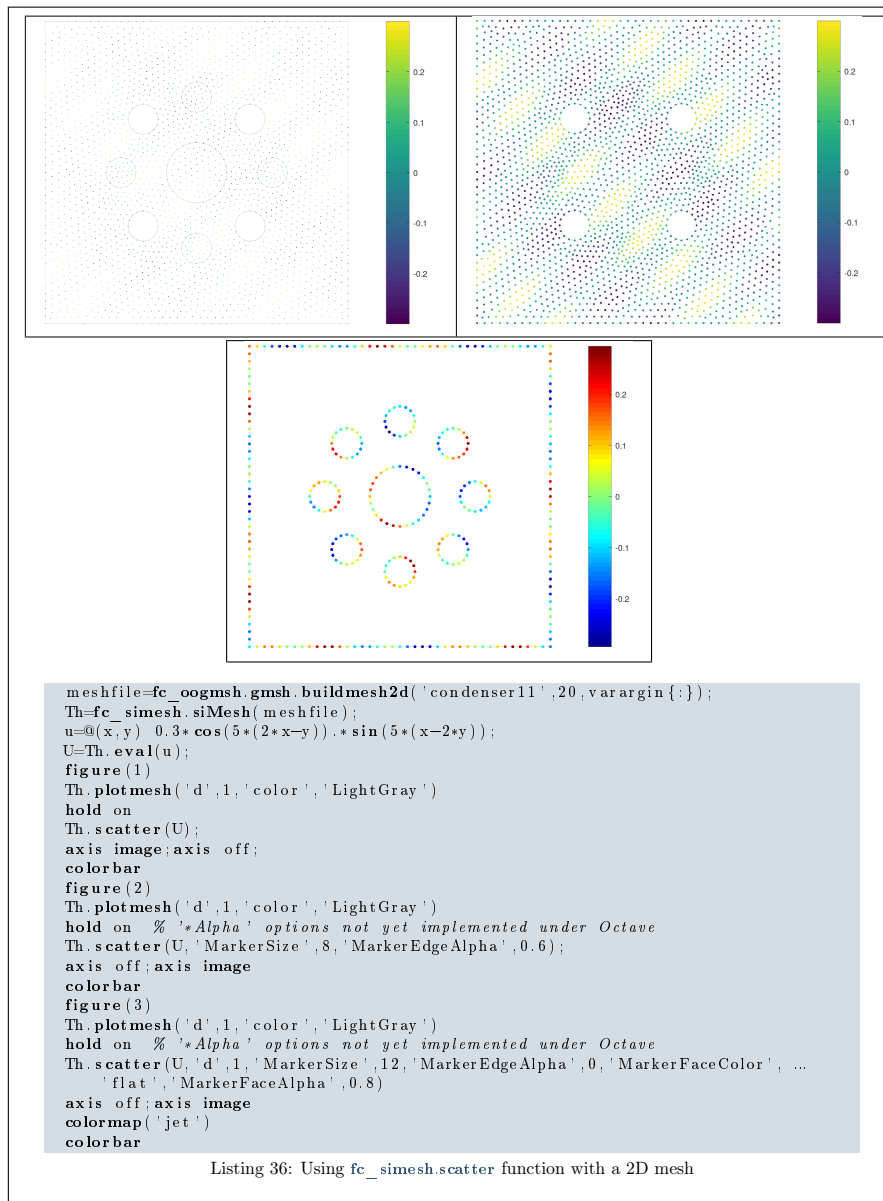
`Th.scatter(u)` displays data `u` on all the  $(Th.d)$ -dimensional simplices elements of `Th`, a `fc_simesh.siMesh` object. The data `u` is an 1D-array of size `Th.nq` or `Th.nqGlobal` or `Th.nqParent`.

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

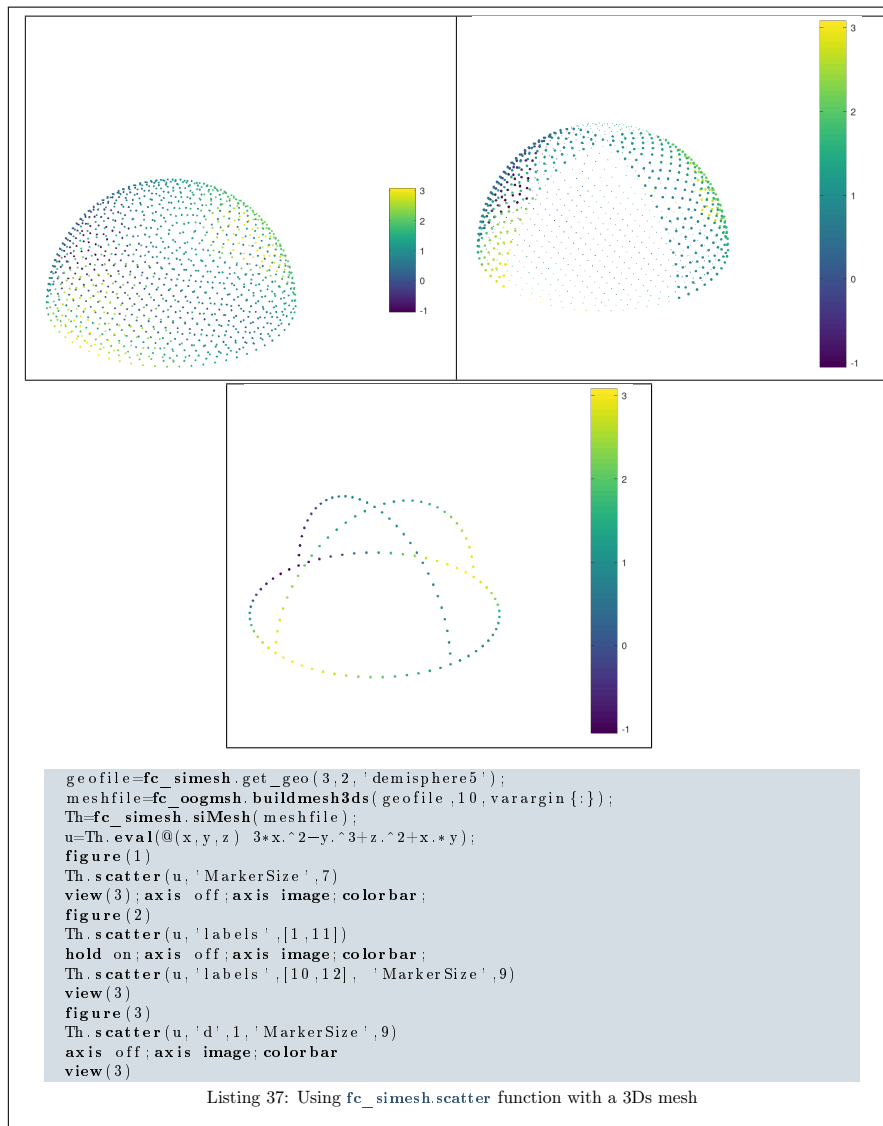
- 'd' : to specify the dimension of the simplices elements (default : Th.d)
- 'labels' : to select the labels of the elements to display data,
- 'MarkerSize' : size of the marker. Default is 1.
- 'ForcePatch' : if **true**, uses **patch** function, otherwise uses **scatter** function in dimension 2 or **scatter3** function in dimension 3. Default is **true**.

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

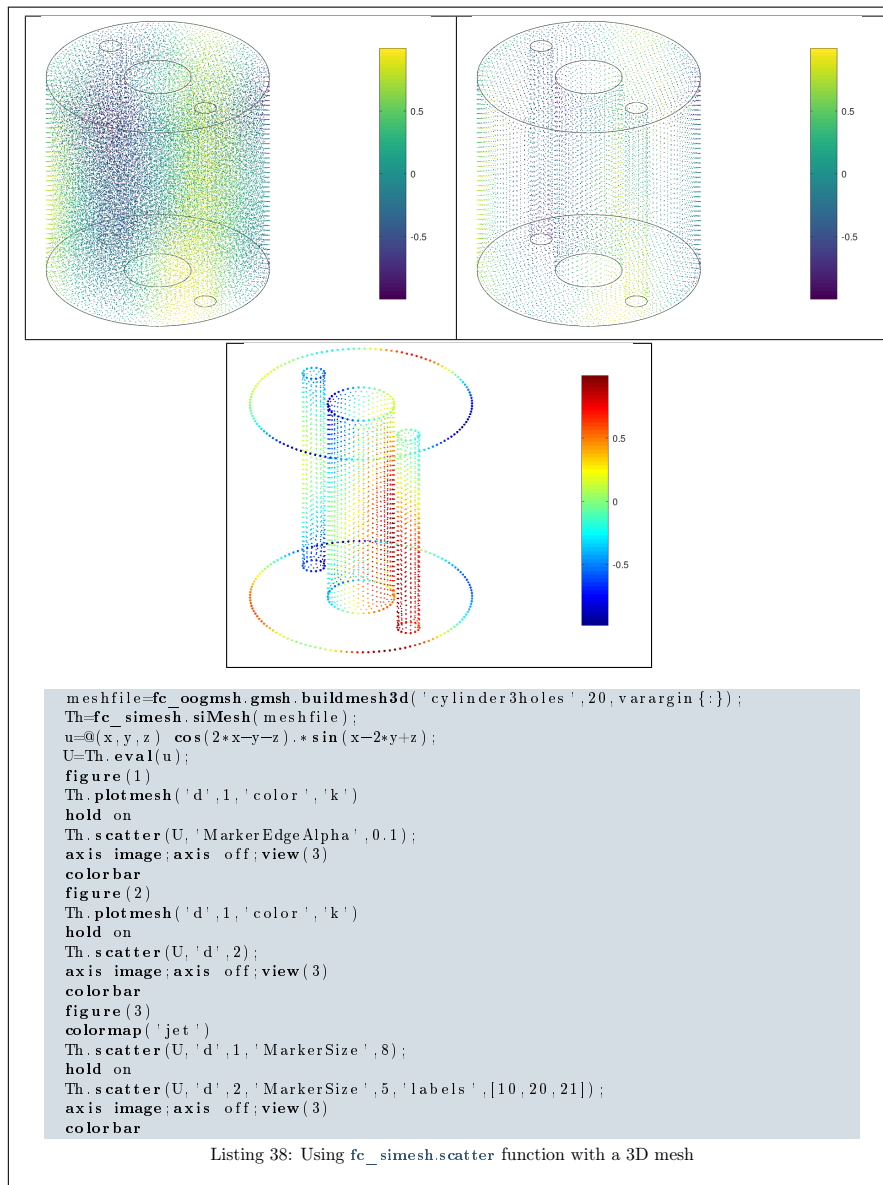
**2D example :** the following code is part of the `fc_simesh.demos.scatter2D` function.



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



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



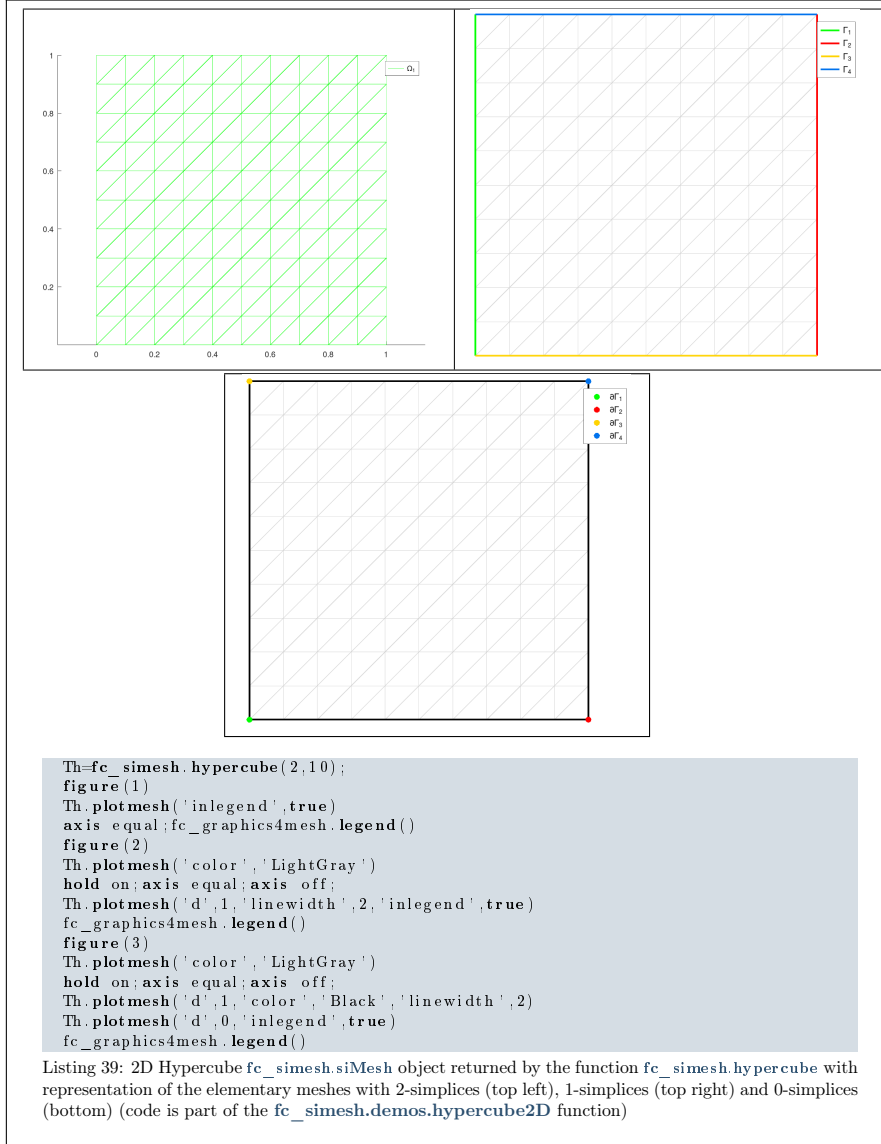
### 3.5 Hypercube as a `fc_simesh.siMesh` object

The function `fc_simesh.hypercube` allows to create a `fc_simesh.siMesh` object representing an hypercube in any dimension. It uses the **FC-HYPERMESH** Octave package.

- `Th=fc_simesh.hypercube(dim,N)` : return a `fc_simesh.siMesh` object representing an hypercube in dimension `dim` and ...
- `Th=fc_simesh.hypercube(dim,N,Key,Value,...)` :

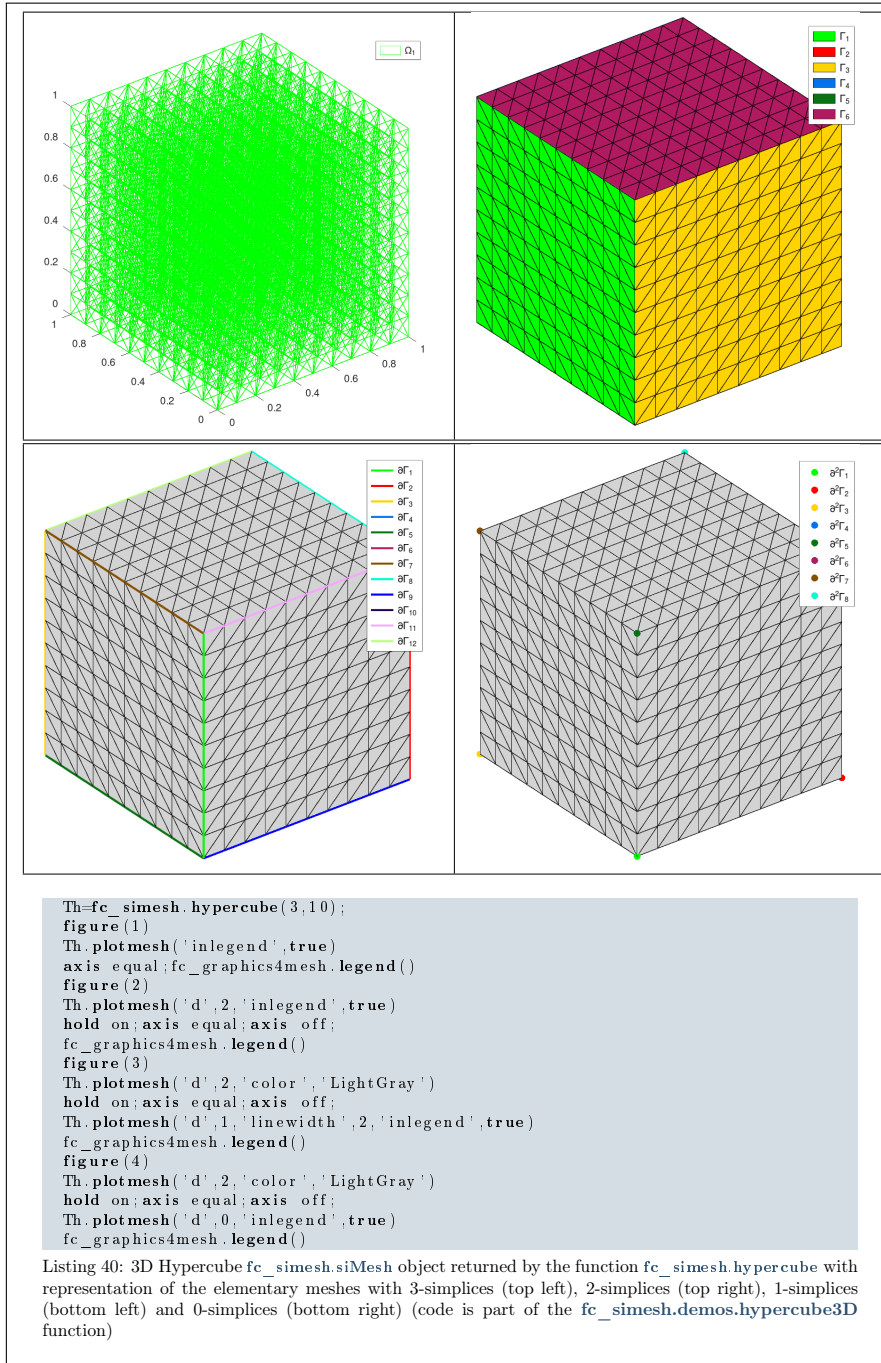
### 3.5.1 2D hypercube

In Listing 1 a usage example generating a 2D hypercube as a `fc_simesh.siMesh` object is given with representation of its elementary meshes.



### 3.5.2 3D hypercube

In Listing 1 a usage example generating a 3D hypercube as a `fc_simesh.siMesh` object is given with representation of its elementary meshes.



### 3.5.3 4D hypercube

In Listing 41 a usage example generating a 4D hypercube as a `fc_simesh.siMesh` object is given.



Listing 41: : function `fc_simesh.hypercube`

```
Th=fc_simesh.hypercube(4,10) ;
disp(Th)
```

Output

```
fc_simesh.siMesh with properties:
    bbox: [ 0 1 0 1 0 1 0 1 ] (1x8 double)
        d: 4 double
        dim: 4 double
        nq: 14641 double
    nqParents: 14641 double
    nsTh: 81 double
    other: []
    sTh: (1x81 cell)
    sThPhysicalTags: []
    sThboundlabs: []
    sThcolors: (81x3 double)
    sThgeolab: []
    sThlab: (1x81 double)
    sThpartlabs: []
    sThphyslab: 1 double
    sThsimp: (1x81 double)
    toGlobal: (1x14641 double)
    toParent: (1x14641 double)
    toParents: (1x1 cell)
```

### 3.5.4 5D hypercube

In Listing 42 a usage example generating a 5D hypercube as a `fc_simesh.siMesh` object is given.

Listing 42: : function `siMesh.HyperCube`

```
Th=fc_simesh.hypercube(5,6) ;
disp(Th)
```

Output

```
fc_simesh.siMesh with properties:
    bbox: [ 0 1 0 1 0 1 0 1 0 1 ] (1x10 double)
        d: 5 double
        dim: 5 double
        nq: 16807 double
    nqParents: 16807 double
    nsTh: 243 double
    other: []
    sTh: (1x243 cell)
    sThPhysicalTags: []
    sThboundlabs: []
    sThcolors: (243x3 double)
    sThgeolab: []
    sThlab: (1x243 double)
    sThpartlabs: []
    sThphyslab: 1 double
    sThsimp: (1x243 double)
    toGlobal: (1x16807 double)
    toParent: (1x16807 double)
    toParents: (1x1 cell)
```

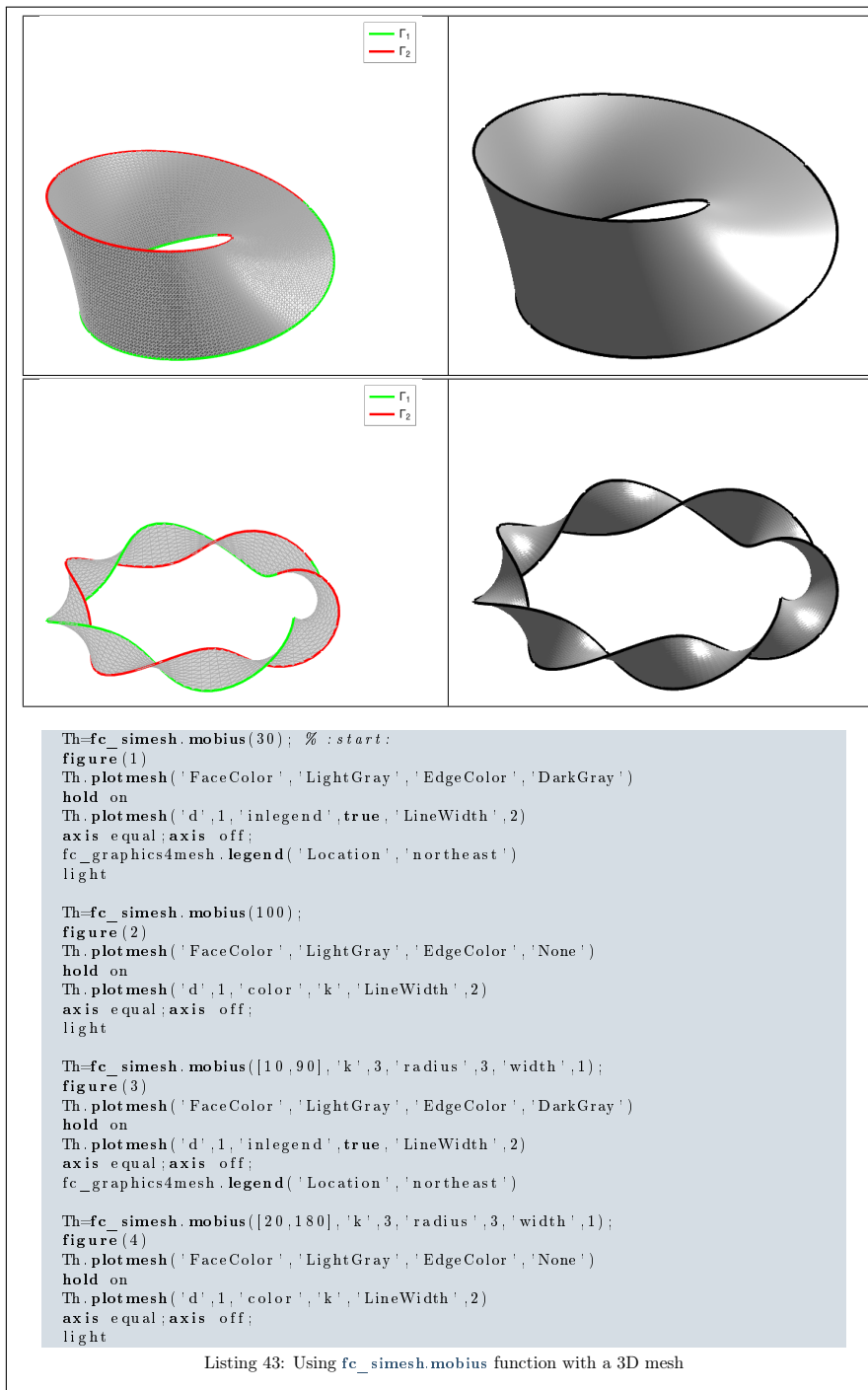
### 3.5.5 Möbius strip as a `fc_simesh.siMesh` object

The function `fc_simesh.mobius` allows to create a `fc_simesh.siMesh` object representing a Möbius strip in dimension 3.

- `Th=fc_simesh.mobius(N)` : return a `fc_simesh.siMesh` object representing a mobius strip. The `N` value is the number of discretisations in  $\theta$  (angle) and  $r$  (radius). if `N` is an array of length 2, then `N(1)` and `N(2)` are respectively the number of discretisations in  $\theta$  and the number of discretisations in  $r$ . If `N` is a scalar then `N` is taken for the both values.

- `Th=fc_simesh.mobius(N,Name,Value,...)` : specifies function options using one or more Name,Value pair arguments,
  - 'radius' : to specify the radius of the mid circle. Default is 1.
  - 'width' : to specify the width of the strip.

**3D example :** the following code is part of the `fc_simesh.demos.mobius3D` function.



# Appendices

## A Listings

|    |                                                                                                                                                                                                                                                                                                                          |    |
|----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 1  | <code>fc_simesh.demos.sample2D01</code> script with figure 1 (top left), figure 2 (top right), figure 3 (bottom left) and figure 4 (bottom right).                                                                                                                                                                       | 3  |
| 2  | : 2D <code>fc_simesh.siMesh</code> object from <code>sample20.geo</code> . . . . .                                                                                                                                                                                                                                       | 8  |
| 3  | : 3D Mesh from <code>quart_sphere2.geo</code> . . . . .                                                                                                                                                                                                                                                                  | 10 |
| 4  | : 3D surface Mesh from <code>demisphere4surf.geo</code> . . . . .                                                                                                                                                                                                                                                        | 12 |
| 5  | : <code>fc_simesh.siMesh</code> constructor . . . . .                                                                                                                                                                                                                                                                    | 14 |
| 6  | : <code>fc_simesh.siMesh</code> find method samples . . . . .                                                                                                                                                                                                                                                            | 15 |
| 7  | : <code>feval</code> method, four ways to defined a function . . . . .                                                                                                                                                                                                                                                   | 16 |
| 8  | : <code>feval</code> method with a vector-valued function . . . . .                                                                                                                                                                                                                                                      | 16 |
| 9  | : <code>eval</code> method, four ways to defined a function . . . . .                                                                                                                                                                                                                                                    | 17 |
| 10 | : <code>eval</code> method with a vector-valued function . . . . .                                                                                                                                                                                                                                                       | 17 |
| 11 | : <code>get_mesh</code> method, four ways to defined a function . . . . .                                                                                                                                                                                                                                                | 18 |
| 12 | : <code>get_nme</code> method . . . . .                                                                                                                                                                                                                                                                                  | 19 |
| 13 | : <code>get_nqe</code> method . . . . .                                                                                                                                                                                                                                                                                  | 19 |
| 14 | : <code>get_labels</code> method . . . . .                                                                                                                                                                                                                                                                               | 20 |
| 15 | Using <code>fc_simesh.moveArray</code> function with a 2D mesh . . . . .                                                                                                                                                                                                                                                 | 21 |
| 16 | Using <code>fc_simesh.moveCell</code> function with a 2D mesh . . . . .                                                                                                                                                                                                                                                  | 22 |
| 17 | Using <code>fc_simesh.moveArray</code> function with a 3Ds mesh . . . . .                                                                                                                                                                                                                                                | 23 |
| 18 | Using <code>fc_simesh.moveCell</code> function with a 3Ds mesh . . . . .                                                                                                                                                                                                                                                 | 24 |
| 19 | Using <code>fc_simesh.moveArray</code> function with a 3D mesh . . . . .                                                                                                                                                                                                                                                 | 25 |
| 20 | Using <code>fc_simesh.moveCell</code> function with a 3D mesh . . . . .                                                                                                                                                                                                                                                  | 26 |
| 21 | Using <code>fc_simesh.plotmesh</code> function with a 2D mesh . . . . .                                                                                                                                                                                                                                                  | 28 |
| 22 | Using <code>fc_simesh.plotmesh</code> function with a 3Ds mesh . . . . .                                                                                                                                                                                                                                                 | 29 |
| 23 | Using <code>fc_simesh.plotmesh</code> function with a 3D mesh . . . . .                                                                                                                                                                                                                                                  | 30 |
| 24 | Using <code>fc_simesh.plot</code> function with a 2D mesh . . . . .                                                                                                                                                                                                                                                      | 32 |
| 25 | Using <code>fc_simesh.plot</code> function with a 3Ds mesh . . . . .                                                                                                                                                                                                                                                     | 33 |
| 26 | Using <code>fc_simesh.plot</code> function with a 3D mesh . . . . .                                                                                                                                                                                                                                                      | 34 |
| 27 | Using <code>fc_simesh.plotiso</code> function with a 2D mesh . . . . .                                                                                                                                                                                                                                                   | 36 |
| 28 | Using <code>fc_simesh.plotiso</code> function with a 3Ds mesh . . . . .                                                                                                                                                                                                                                                  | 37 |
| 29 | Using <code>fc_simesh.plotiso</code> function with a 3D mesh . . . . .                                                                                                                                                                                                                                                   | 38 |
| 30 | 3D mesh : <code>slicemesh</code> method . . . . .                                                                                                                                                                                                                                                                        | 39 |
| 31 | 3D mesh : <code>slice</code> method . . . . .                                                                                                                                                                                                                                                                            | 40 |
| 32 | 3D mesh : <code>sliceiso</code> method . . . . .                                                                                                                                                                                                                                                                         | 42 |
| 33 | Using <code>fc_simesh.plotquiver</code> function with a 2D mesh . . . . .                                                                                                                                                                                                                                                | 44 |
| 34 | Using <code>fc_simesh.plotquiver</code> function with a 3Ds mesh . . . . .                                                                                                                                                                                                                                               | 45 |
| 35 | Using <code>fc_simesh.plotquiver</code> function with a 3D mesh . . . . .                                                                                                                                                                                                                                                | 46 |
| 36 | Using <code>fc_simesh.scatter</code> function with a 2D mesh . . . . .                                                                                                                                                                                                                                                   | 48 |
| 37 | Using <code>fc_simesh.scatter</code> function with a 3Ds mesh . . . . .                                                                                                                                                                                                                                                  | 49 |
| 38 | Using <code>fc_simesh.scatter</code> function with a 3D mesh . . . . .                                                                                                                                                                                                                                                   | 50 |
| 39 | 2D Hypercube <code>fc_simesh.siMesh</code> object returned by the function <code>fc_simesh.hypercube</code> with representation of the elementary meshes with 2-simplices (top left), 1-simplices (top right) and 0-simplices (bottom) (code is part of the <code>fc_simesh.demos.hypercube2D</code> function) . . . . . | 51 |

|    |                                                                                                                                                                                                                                                                                                                                                           |    |
|----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 40 | 3D Hypercube <code>fc_simesh.siMesh</code> object returned by the function <code>fc_simesh.hypercube</code> with representation of the elementary meshes with 3-simplices (top left), 2-simplices (top right), 1-simplices (bottom left) and 0-simplices (bottom right) (code is part of the <code>fc_simesh.demos.hypercube3D</code> function) . . . . . | 52 |
| 41 | : function <code>fc_simesh.hypercube</code> . . . . .                                                                                                                                                                                                                                                                                                     | 53 |
| 42 | : function <code>siMesh.HyperCube</code> . . . . .                                                                                                                                                                                                                                                                                                        | 53 |
| 43 | Using <code>fc_simesh.mobius</code> function with a 3D mesh . . . . .                                                                                                                                                                                                                                                                                     | 55 |

## B References

- [1] F. Cuvelier. `fc_oogmsh`: an object-oriented Octave package to run `gmsh` and read mesh files. <http://www.math.univ-paris13.fr/~cuvelier/software/>, 2017. User’s Guide.
- [2] F. Cuvelier. `fc_siplt`: an add-on to the `fc_simesh` Octave package for displaying simplices meshes or datas on simplices meshes. <http://www.math.univ-paris13.fr/~cuvelier/software/>, 2017. User’s Guide.
- [3] F. Cuvelier. `fc_hypermesh`: a object-oriented Octave package to mesh any d-orthotopes (hyperrectangle in dimension d) and their m-faces with high order simplices or orthotopes. <http://www.math.univ-paris13.fr/~cuvelier/software/>, 2019. User’s Guide.

# Informations for git maintainers of the



# Octave package

git informations on the packages used to build this manual

```
-----
name : fc-simesh
tag : 0.4.5
commit : 2435c8c22b6280df5521be7d137c3a206f3d455d
date : 2022-12-21
time : 14-27-51
status : 0
-----
name : fc-tools
tag : 0.0.34
commit : 625f31d5dd97131d3f3a96ef415f810c008c19c6
date : 2023-01-04
time : 12-59-08
status : 0
-----
name : fc-bench
tag : 0.1.3
commit : 6c8969bb49b90c7cae34e70c88dd2f968766376b
date : 2022-12-17
time : 13-58-43
status : 0
-----
name : fc-hypermesh
tag : 1.0.4
commit : 7f56fea428e57d2f30ae9d7b1f99bd6078807d23
date : 2022-12-19
time : 09-35-48
status : 0
-----
name : fc-amat
tag : 0.1.3
commit : 90dbaa2839188cf8e01629817e5f8ba1b7188052
date : 2022-12-19
time : 07-41-33
status : 0
-----
name : fc-meshtools
tag : 0.1.4
commit : 292552bae1b8dd1f3d58e73c68aca17999f68e2
date : 2022-12-21
time : 11-38-20
status : 0
-----
name : fc-graphics4mesh
tag : 0.1.5.a
commit : b994b7b2a57399848541708cae3c33f79787cb1a
date : 2022-12-19
time : 08-28-40
status : 0
-----
name : fc-oogmsh
tag : 0.2.4.a
commit : 8738562fc0b1cbdb028a8b58c5ba4f019e452831
date : 2022-12-21
time : 14-42-29
status : 0
-----
name : fc-siplt
tag : 0.2.5
commit : b4d6a9b05dcd6ac643d7235a057f89704c7e72a
date : 2022-12-21
time : 13-26-51
status : 0
-----
[fc-tools] waiting 2(s) to finish saving figures
```

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

```
-----  
name : fctools  
tag :  
commit : c9a33ce7b4daf90f66e5e49856e69afaldac0a3  
date : 2022-12-17  
time : 07:57:20  
status : 1  
-----  
[fc-tools] waiting 2(s) to finish saving figures
```

Using the remote configuration repository:

```
url      ssh://lagagit/MCS/Cuvelier/Matlab/fc-config  
commit   268d29786e42ee37b3531aed5a97635dfc1d9ada
```