



Octave package, User's Guide* version 0.4.1

François Cuvelier[†]

March 8, 2020

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  package is used.

0 Contents

1	Introduction	2
2	Installation	4
3	Mesh Objects	6
3.1	<code>fc_simesh.siMeshElt</code> object	6
3.2	<code>fc_simesh.siMesh</code> object	8
3.3	Mesh samples	9
3.3.1	2-simplicial mesh in $\mathbb{R}^2$	9
3.3.2	Sample of a 3-simplicial mesh in $\mathbb{R}^3$	11

* $\text{\LaTeX}$ manual, revision 0.4.1, compiled with Octave 5.2.0, and packages `fc-simesh`[0.4.1], `fc-tools`[0.0.31], `fc-bench`[0.1.2], `fc-hypermesh`[1.0.3], `fc-amat`[0.1.2], `fc-meshtools`[0.1.3], `fc-graphics4mesh`[0.1.2], `fc-oogmsh`[0.2.3], `fc-siplt`[0.2.1]

[†]Université Sorbonne Paris Nord, LAGA, CNRS, UMR 7539, F-93430, Villetaneuse, France, cuvelier@math.univ-paris13.fr.

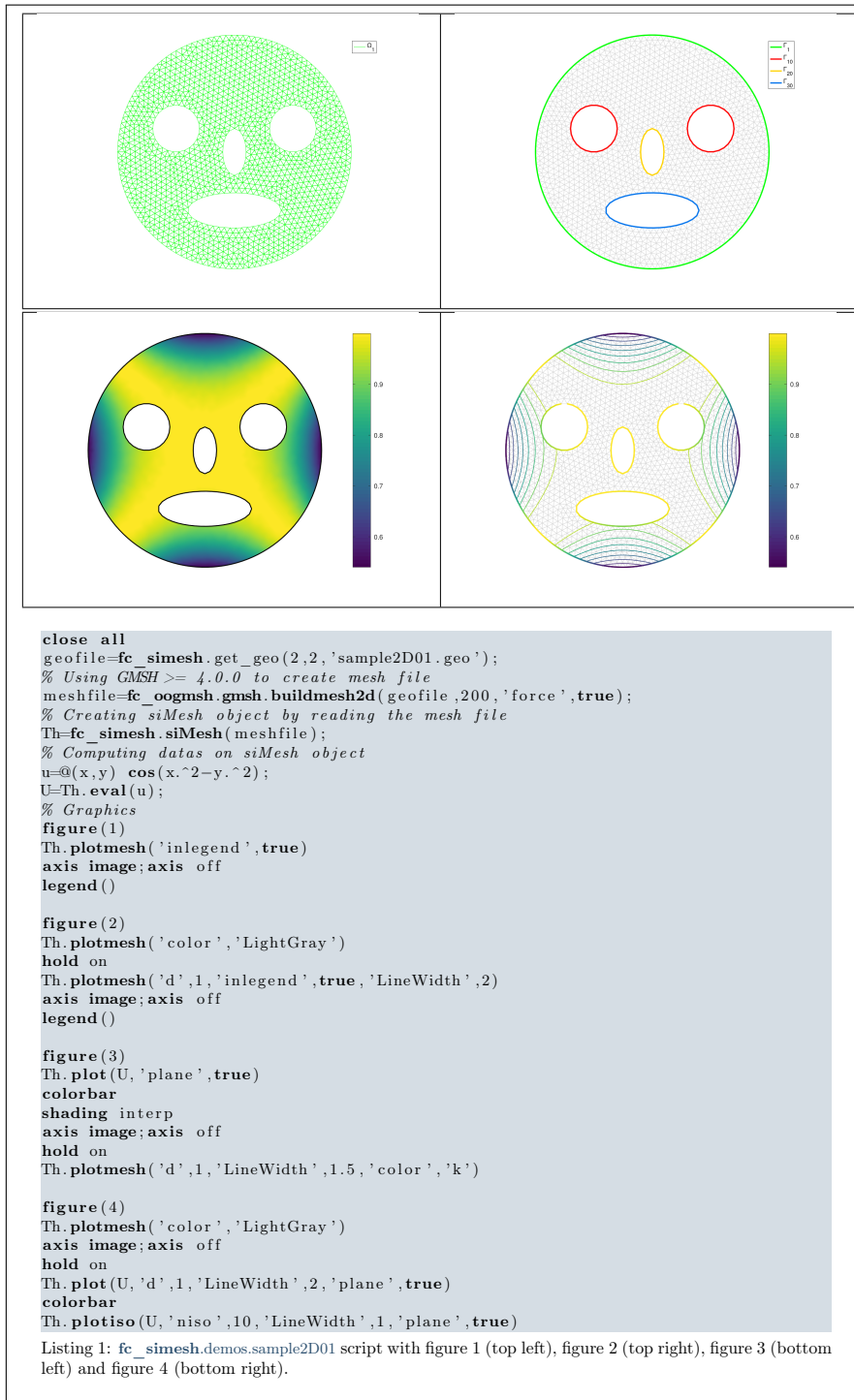
This work was supported by the ANR project DEDALES under grant ANR-14-CE23-0005.

3.3.3	Sample of a 2-simplicial mesh in $\mathbb{R}^3$	13
3.4	Methods of the <code>fc_simesh.siMesh</code> object	14
3.4.1	<code>fc_simesh.siMesh</code> constructor	14
3.4.2	<code>find</code> method	15
3.4.3	<code>feval</code> method	16
3.4.4	<code>eval</code> method	17
3.4.5	<code>get_h</code> method	19
3.4.6	<code>get_mesh</code> method	19
3.4.7	<code>get_nme</code> method	19
3.4.8	<code>get_nq</code> method	20
3.4.9	<code>plotmesh</code> method	21
3.4.10	<code>plot</code> method	25
3.4.11	<code>plotiso</code> method	29
3.4.12	<code>slicemesh</code> method	33
3.4.13	<code>slice</code> method	34
3.4.14	<code>sliceiso</code> method	35
3.4.15	<code>plotquiver</code> method	36
3.5	Hypercube as a <code>fc_simesh.siMesh</code> object	39
3.5.1	2D hypercube	39
3.5.2	3D hypercube	40
3.5.3	4D hypercube	41
3.5.4	5D hypercube	42

Appendices	43
------------	----

1 Introduction

The `fc_simesh` 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 `oogmsh` 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 `siplt` 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 [hypermesh](#) package[3] can be used to obtain meshes of an hypercube by using the `fc_simesh.hypercube` function.

2 Installation

This package was tested on various OS with Octave releases:

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

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

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

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  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
>> mfc_simesh_install
```

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

```

Parts of the <fc-simesh> Octave package.
Copyright (C) 2017-2010 F. Cuvelier

1- Downloading and extracting the packages
2- Setting the <fc-simesh> package
Write in ~/Octave/fc-simesh-full/fc_simesh-0.4.1/configure_loc.m ...
3- Using packages :
    ->          fc-tools : 0.0.31
    ->          fc-bench : 0.1.2
    ->          fc-hypermesh : 1.0.3
    ->          fc-amat : 0.1.2
    ->          fc-meshtools : 0.1.3
    ->          fc-graphics4mesh : 0.1.2
    ->          fc-oogmsh : 0.2.3
    ->          fc-siplt : 0.2.1
    with        fc-simesh : 0.4.1
*** Using instructions
    To use the <fc-simesh> package:
    addpath('~/Octave/fc-simesh-full/fc_simesh-0.4.1')
    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-0.4.1')
>> 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-0.0.31/configure_loc.m ...
Try to use default parameters!
  Use fc_bench.configure to configure.
Write in ~/Octave/fc-simesh-full/fc_bench-0.1.2/configure_loc.m ...
Try to use default parameters!
  Use fc_hypermesh.configure to configure.
Write in ~/Octave/fc-simesh-full/fc_hypermesh-1.0.3/configure_loc.m ...
Try to use default parameters!
  Use fc_amat.configure to configure.
Write in ~/Octave/fc-simesh-full/fc_amat-0.1.2/configure_loc.m ...
Try to use default parameters!
  Use fc_meshtools.configure to configure.
Write in ~/Octave/fc-simesh-full/fc_meshtools-0.1.3/configure_loc.m ...
Try to use default parameters!
  Use fc_graphics4mesh.configure to configure.
Write in ...
  ~/Octave/fc-simesh-full/fc_graphics4mesh-0.1.2/configure_loc.m ...
Try to use default parameters!
  Use fc_oogmsh.configure to configure.
Write in ~/Octave/fc-simesh-full/fc_oogmsh-0.2.3/configure_loc.m ...
Configured to use gmsh 4.5.1 with default MSH file format version 4.1
Try to use default parameters!
  Use fc_siplt.configure to configure.
Write in ~/Octave/fc-simesh-full/fc_siplt-0.2.1/configure_loc.m ...
Using fc_simesh[0.4.1] with fc_tools[0.0.31], fc_bench[0.1.2], ...
      fc_hypermesh[1.0.3],
                        fc_amat[0.1.2], fc_meshtools[0.1.3], ...
                        fc_graphics4mesh[0.1.2],
                        fc_oogmsh[0.2.3], fc_siplt[0.2.1].

```

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

```

Configured to use gmsh 4.5.1 with default MSH file format version 4.1
Using fc_simesh[0.4.1] with fc_tools[0.0.31], fc_bench[0.1.2], ...
    fc_hypermesh[1.0.3],
                                fc_amat[0.1.2], fc_meshtools[0.1.3], ...
                                fc_graphics4mesh[0.1.2],
                                fc_oogmsh[0.2.3], fc_siplt[0.2.1].

```

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 $\mathbb{R}^{\text{dim}}$, $0 \leq k \leq \text{dim}$, is a polytope which is the convex hull of its $k + 1$ vertices of $\mathbb{R}^{\text{dim}}$. More formally, suppose the $k + 1$ vertices $q^0, \dots, q^k \in \mathbb{R}^{\text{dim}}$ 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 $\mathbb{R}^{\text{dim}}$, $0 \leq k \leq \text{dim}$, a mesh with **unique label** only composed with k -simplices.

A **d -simplicial mesh** in $\mathbb{R}^{\text{dim}}$, $0 \leq d \leq \text{dim}$, 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 space dimension
<code>d</code>	: integer ($0 \leq d \leq \dim$)
<code>nq</code>	: integer number of vertices
<code>nme</code>	: integer number of elements (d -simplices)
<code>q</code>	: $\dim$ -by- <code>nq</code> array of reals array of vertex coordinates
<code>me</code>	: $(d + 1)$ -by- <code>nme</code> array of integers connectivity array for mesh elements
<code>vols</code>	: 1-by- <code>nme</code> array of reals array of mesh element volumes
<code>h</code>	: double mesh step size (=maximum edge length in the mesh)
<code>toGlobal</code>	: 1-by- <code>nq</code> array of integers convert from local to global mesh vertices numbering. Prefer the use of <code>toParents{end}</code> instead. <i>It will be removed in a future release.</i>
<code>toParent</code>	: 1-by- <code>nq</code> array of integers convert from local to parent mesh vertices numbering (same as <code>toGlobal</code> if not part of a partitioned mesh). Prefer the use of <code>toParents{1}</code> instead. <i>It will be removed in a future release.</i>
<code>nqParents</code>	: 1-by- n array of integers <code>nqParents(1)</code> number of vertices in the <i>parent</i> mesh, <code>nqParents(2)</code> number of vertices in the <i>parent</i> of the <i>parent</i> mesh, <code>nqParents(end)</code> number of vertices in the global mesh.
<code>toParents</code>	: 1-by- n cell array <code>toParents{1}</code> indices array which convert local ver- tices numbering to the <i>parent</i> mesh vertices number- ing, <code>toParents{2}</code> indices array which convert local ver- tices numbering to the <i>parent</i> of the <i>parent</i> mesh, <code>toParents{end}</code> indices array which convert local ver- tices 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 space dimension
<code>d</code>	: integer 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 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> label of i -th <code>fc_simesh.siMeshElt</code> object in <code>sTh</code> is number <code>sThlab(i)</code>
<code>nq</code>	: integer number of vertices in the mesh
<code>toGlobal</code>	: 1-by- <code>nq</code> array of integers convert from local to global mesh vertices numbering. Prefer the use of <code>ndtoParent</code> instead. <i>It will be removed in a future release.</i>
<code>toParent</code>	: 1-by- <code>nq</code> array of integers convert from local to parent mesh vertices numbering (same as <code>toGlobal</code> if not part of a partitioned mesh). Prefer the use of <code>toParents1</code> instead. <i>It will be removed in a future release.</i>
<code>nqParents</code>	: 1-by- n array of integers Only used with partitioned mesh and the FC-PSIMESH package.
<code>toParents</code>	: 1-by- n cell array Only used with partitioned mesh and the FC-PSIMESH 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
```

3.3.1 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 4.5.2
[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 4.5.2 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: 2326 double
  nqParents: 2326 double
  nsTh: 11 double
  other: (1x1 struct)
  sTh: (1x11 cell)
  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 ] (1x4 double)
  sThsimp: [ 2 2 2 2 1 1 1 1 1 1 1 ] (1x11 double)
  toGlobal: (1x2326 double)
  toParent: (1x2326 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
  nq: 41 double
  nqGlobal: 2326 double
  nqParent: 2326 double
  nqParents: 2326 double
  order: 1 double
  partlab: []
  q: (2x41 double)
  toGlobal: (1x41 double)
  toParent: (1x41 double)
  toParents: (1x1 cell)
  vols: (1x40 double)
```

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

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

and we note

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

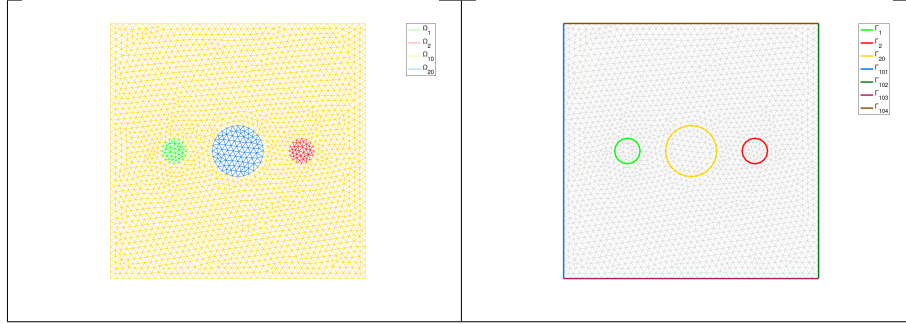


Figure 1: 2D `fc_simesh.siMesh` object from `sample20.geo`

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 : $\Gamma_i \forall i \in \{1, 2, 20, 101, 102, 103, 104\}$

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 4.5.2
[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 4.5.2 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: 1171 double
  nqParents: 1171 double
  nsTh: 23 double
  other: (1x1 struct)
  sTh: (1x23 cell)
  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 ] (1x2 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: (1x1171 double)
  toParent: (1x1171 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: 0.127349 double
  label: 7 double
  me: (3x359 double)
  nme: 359 double
  nq: 203 double
  nqGlobal: 1171 double
  nqParent: 1171 double
  nqParents: 1171 double
  order: 1 double
  partlab: []
  q: (3x203 double)
  toGlobal: (1x203 double)
  toParent: (1x203 double)
  toParents: (1x1 cell)
  vols: (1x359 double)
```

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 : $\Gamma_i \forall i \in \llbracket 1, 7 \rrbracket$
- nine 1-simplicial elementary meshes : $\partial\Gamma_i \forall i \in \llbracket 1, 9 \rrbracket$

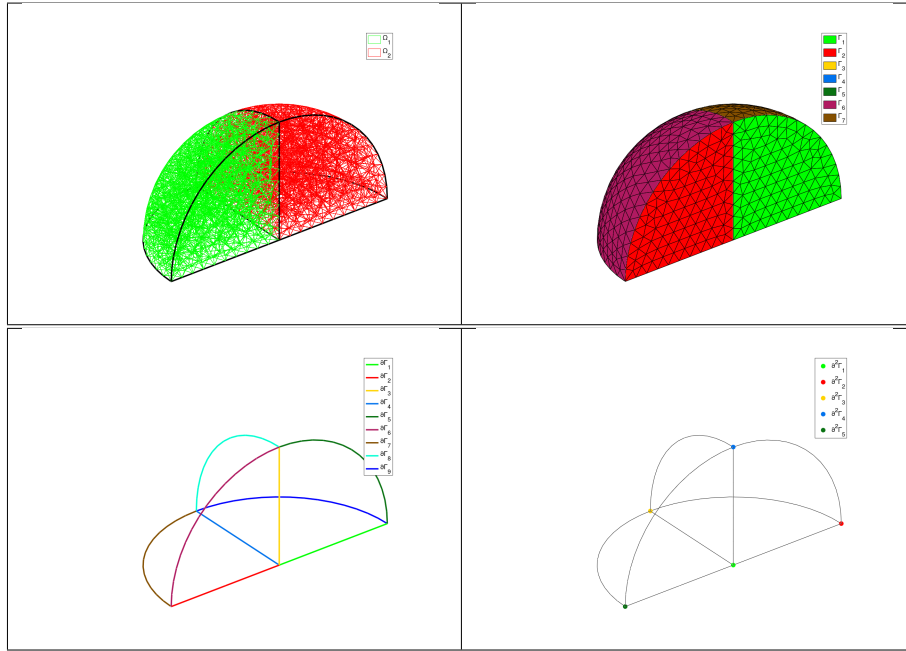


Figure 2: 3D Mesh from `quart_sphere2.geo`

- 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('demicphere4surf',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/demicphere4surf.geo
[fc-oogmsh] Starting building mesh <fc-oogmsh>/meshes/demicphere4surf-5.msh with gmsh 4.5.2
[fc-oogmsh] Using command : gmsh -2 -setnumber N 5 -string "Mesh.MshFileVersion=4.1;" ...
<fc-oogmsh>/geodir/3ds/demicphere4surf.geo -o <fc-oogmsh>/meshes/demicphere4surf-5.msh
Be patient...
[fc-oogmsh] Using gmsh 4.5.2 to write MSH file format version 4.1 in ...
<fc-oogmsh>/meshes/demicphere4surf-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)
  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 ] (1x4 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
  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)
```

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 : $\Gamma_i \forall i \in \llbracket 1, 8 \rrbracket$

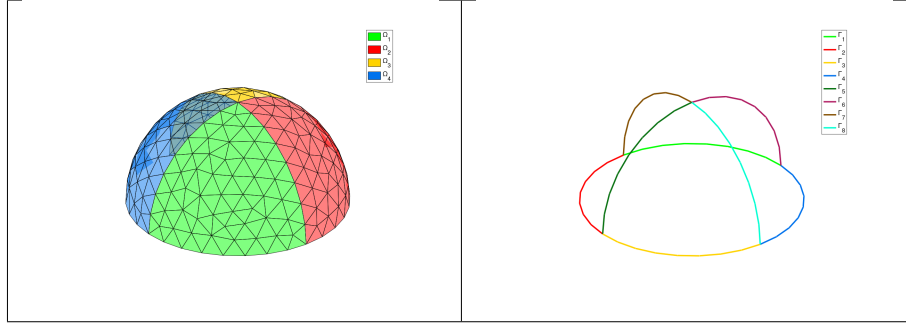


Figure 3: 3D surface Mesh from `demisphere4surf.geo`, label of the domains (left) and label of the boundaries (right)

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)
    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 ] (1x6 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], \n',...
        labels=[%s] \n', num2str(idx), num2str(Th.sThlab(idx)) )
idx=Th.find(2);
fprintf('2-simplices \siMeshElt \n \indices: \[%s], \n',...
        labels=[%s] \n', num2str(idx), num2str(Th.sThlab(idx)) )
idx=Th.find(2,4);
fprintf('2-simplices \siMeshElt \n \with_label==4 \n \indices: \[%s], \n',...
        labels=[%s] \n', num2str(idx), num2str(Th.sThlab(idx)) )
idx=Th.find(2,[6,4,2,10]);
fprintf('2-simplices \siMeshElt \n \with_label in [6,4,2,10] \n \indices: \[%s], \n',...
        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: 1171 double
    nqParents: 1171 double
    nsTh: 23 double
    other: (1x1 struct)
    sTh: (1x23 cell)
    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 ] (1x2 double)
    sThsimp: [ 3 3 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 0 0 0 0 0 ] (1x23 double)
    toGlobal: (1x1171 double)
    toParent: (1x1171 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: [4 6 8], labels=[2 4 6]

```

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)
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 ] (1x6 double)
    sThsimp: [ 2 2 2 2 2 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 : 4784
Number of 2-simplices : 1651
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 : 1171
Number of vertices in 2-simplices elementary meshes : 811
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 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

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

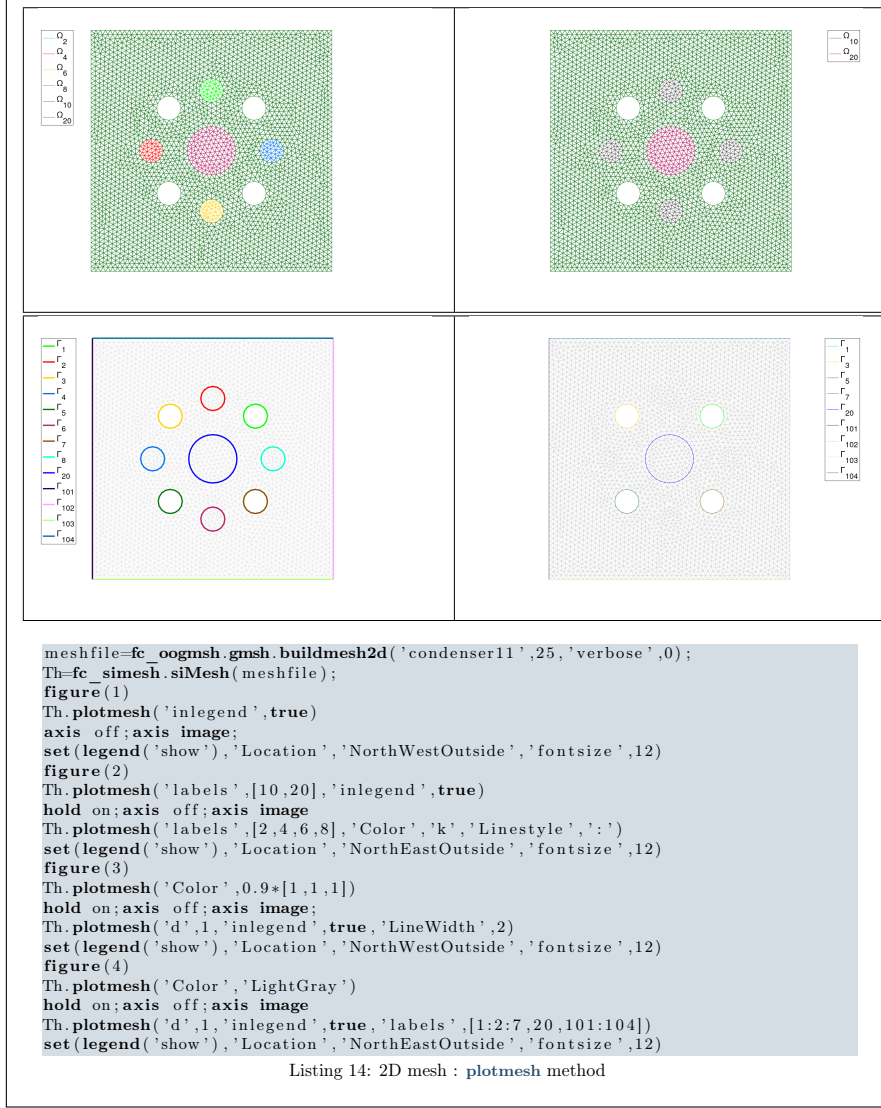
The normal vector $P(i, 1 : 3)$ pointed to the part of the mesh not displayed. (only for simplices in dimension 3) default : `[]` (no cut).

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

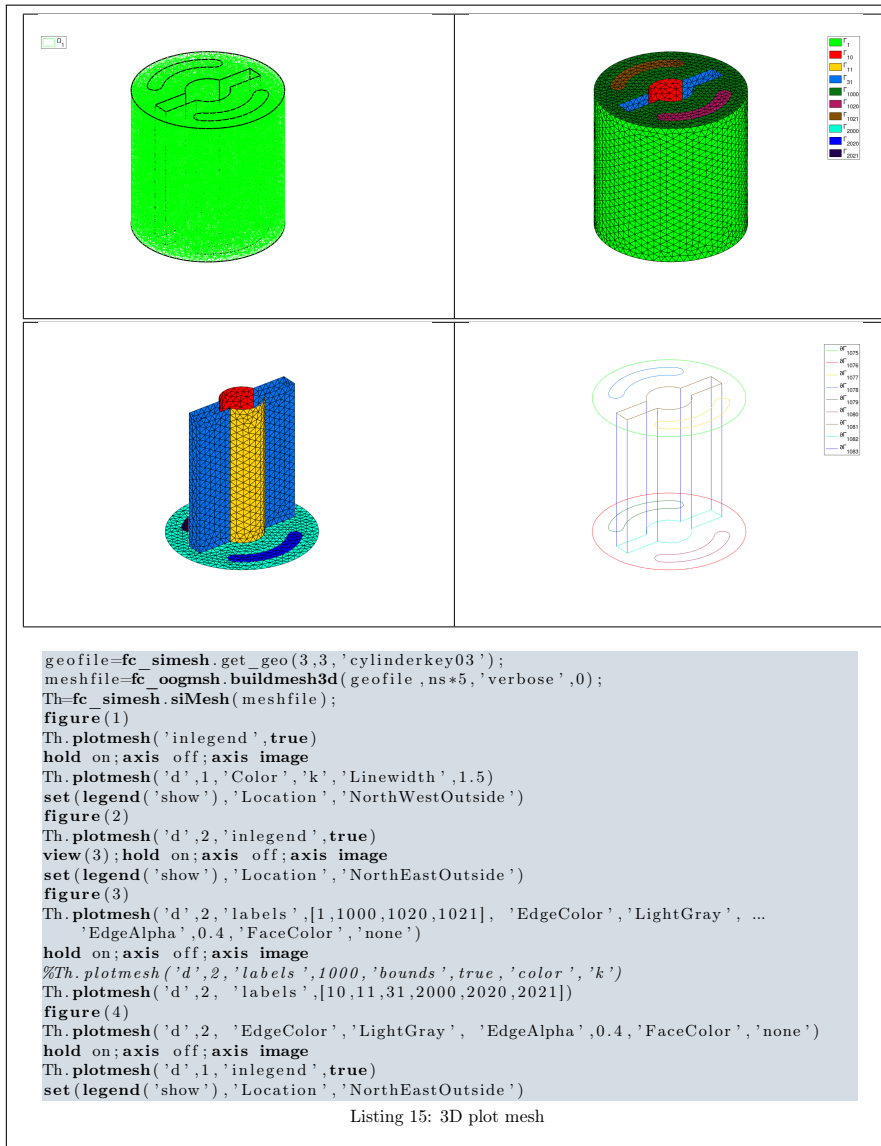
One can use any option of the following functions according to the type of d -simplex to be represented.

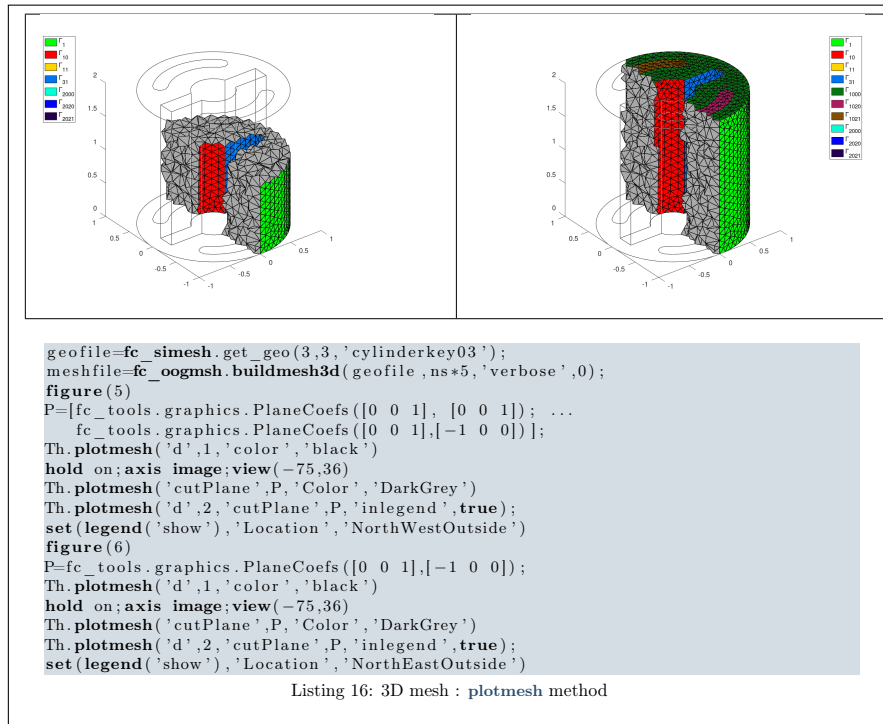
- In dimension 3,
 - if $d == 3$, `patch` function is used,
 - if $d == 2$, `trimesh` function is used,
 - if $d == 1$, `plot3` function is used,
 - if $d == 0$, `plot3` function is used,
- In dimension 2,
 - if $d == 2$, `trimesh` function is used,
 - if $d == 1$, `plot` function is used,
 - if $d == 0$, `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 example use the *.geo* file `condenser11.geo` which is in the directory `geodir` of the toolbox

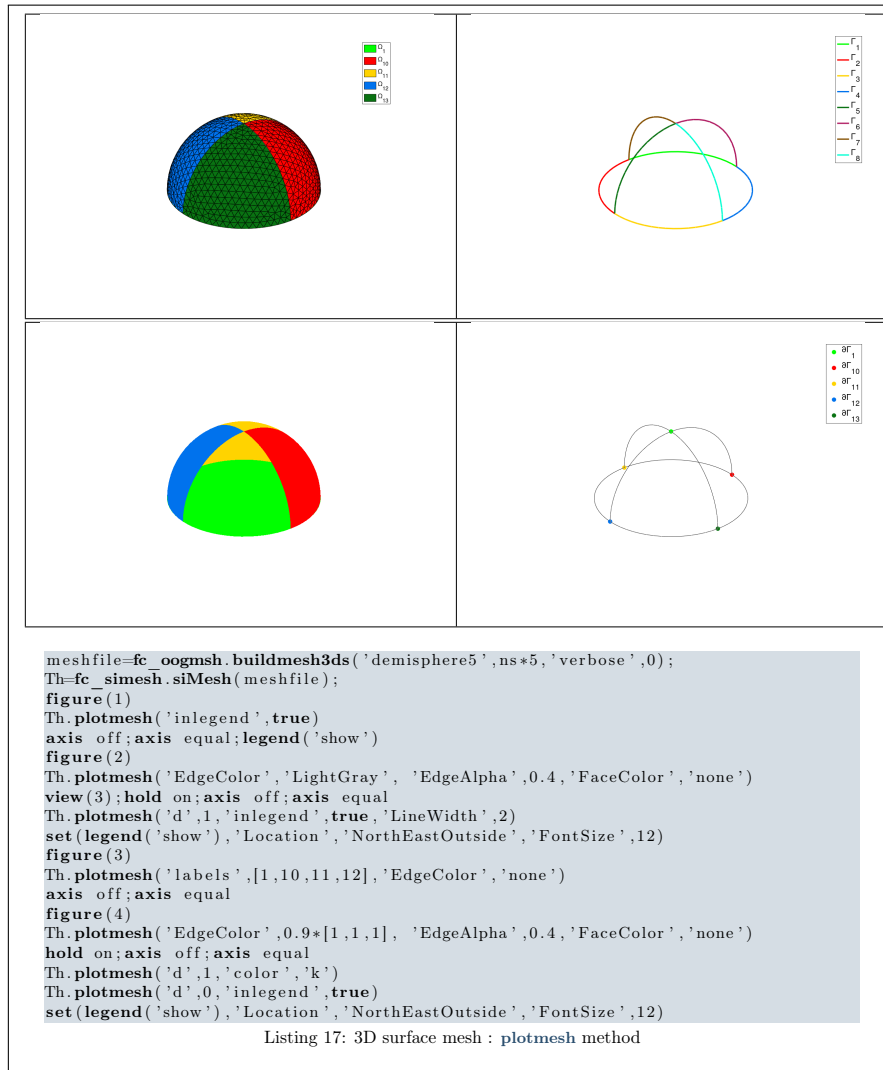


3D example The following example use the *.geo* file `cylinderkey.geo` which is in the directory `geodir` of the toolbox. This file contains description of a 3D mesh with simplices of dimensions 1, 2 and 3.





3D surface example The following example use the *.geo* file `demisphere5.geo` which is in the directory `geodir` of the toolbox. This file contains description of a 3D surface mesh with simplices of dimensions 1 and 2.



3.4.10 `plot` method

The `plot` method displays scalar datas on the mesh or parts of the mesh defined by an `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

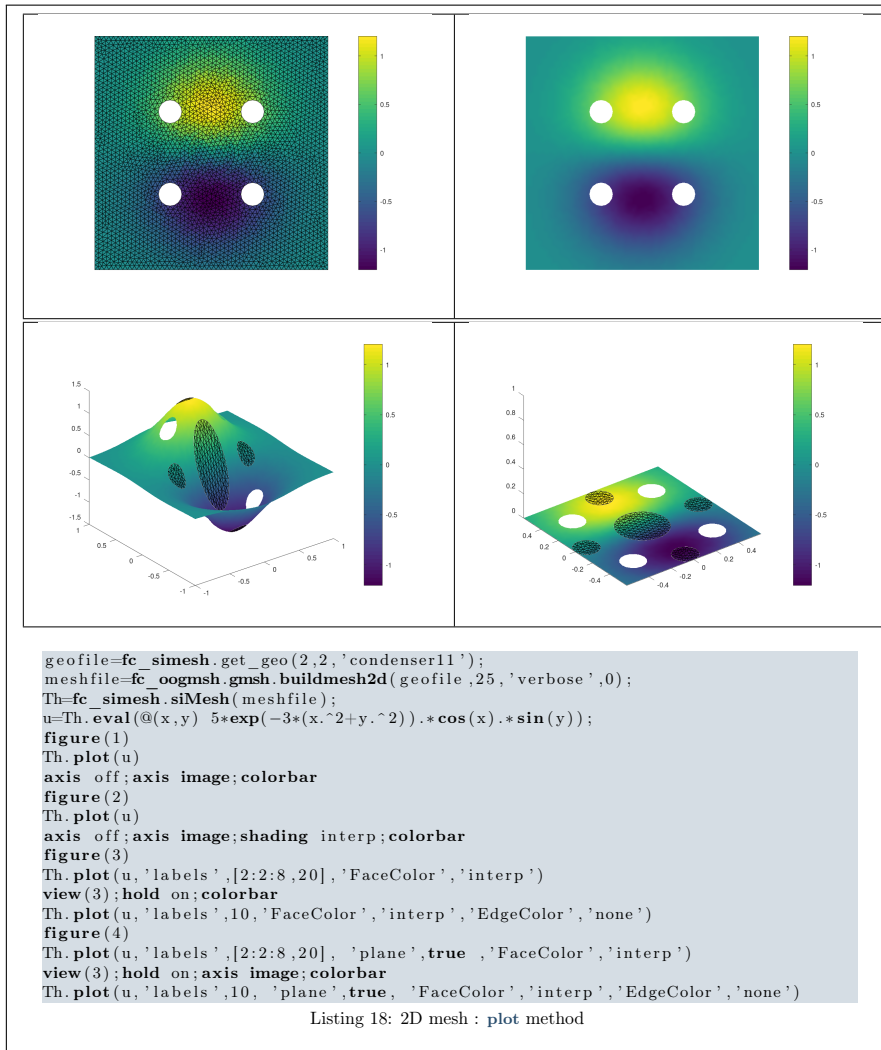
- `'d'` : to specify the dimension of the simplices elements (default : `Th.d`)
- `'labels'` : to select the labels of the elements to display data,
- `'plane'` : if true, made a 2D representation in the xy -plane, otherwise made a 3D representation with z -value set to `u` (default : `false`)

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

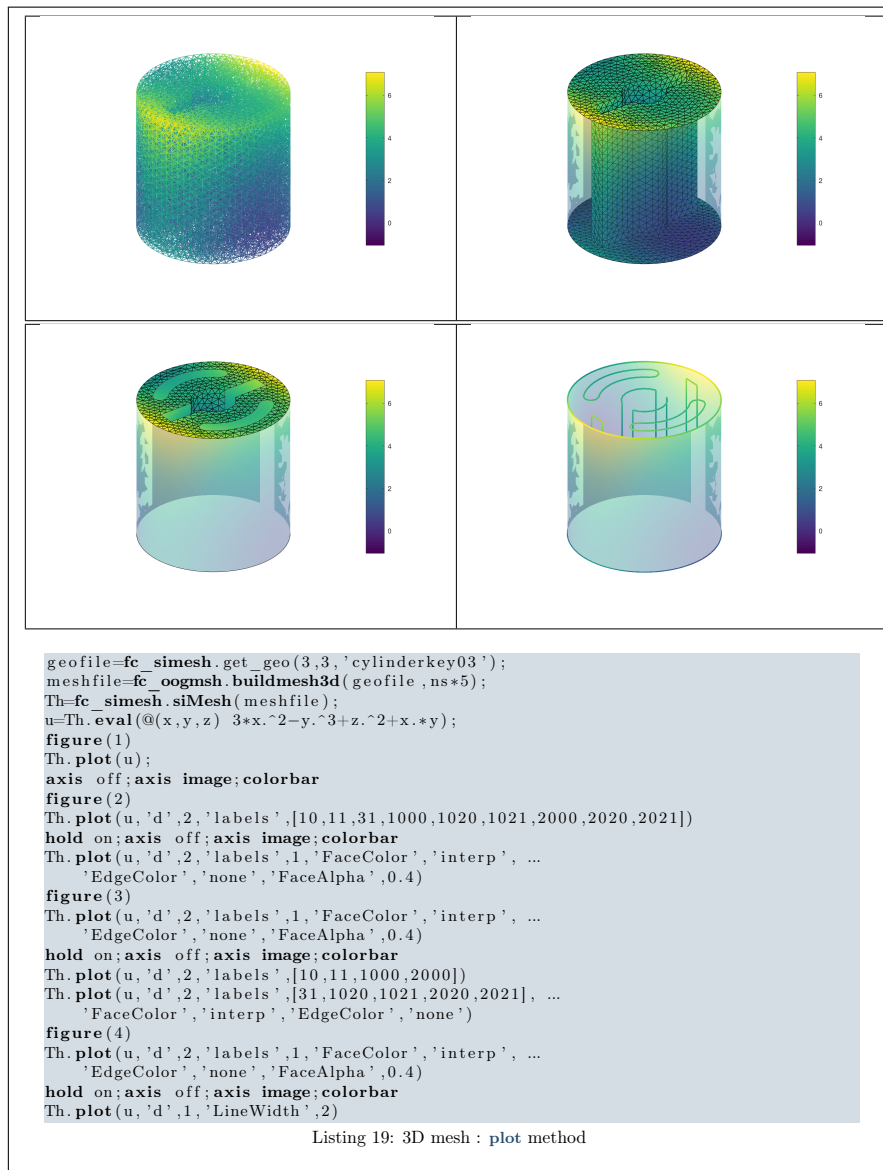
One can use any option of the following functions according to the type of d -simplex.

- 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

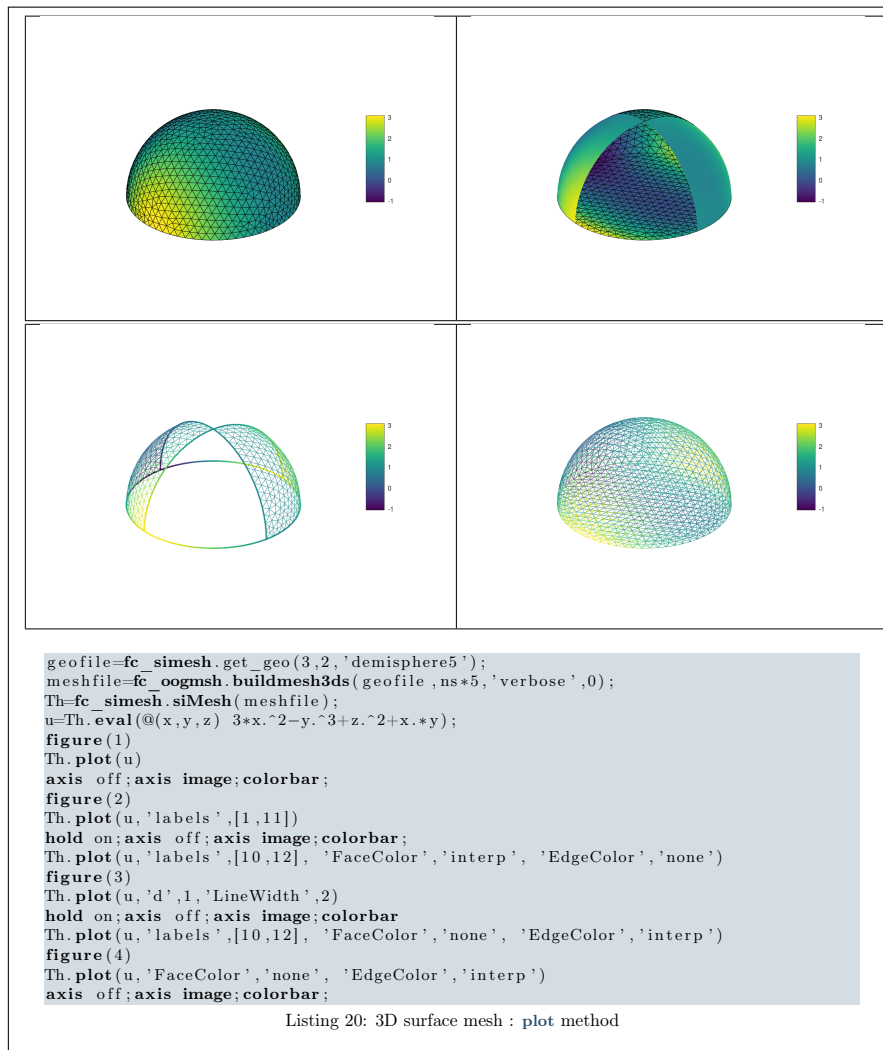
2D example The following example use the `.geo` file `condenser11.geo` which is in the directory `geodir` of the toolbox.



3D example The following example use the *.geo* file *cylinderkey.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.



3D surface example The following example use the *.geo* file *demisphere5.geo* which is in the directory *geodir* of the toolbox. This file contains description of a 3D surface mesh with simplices of dimensions 1 and 2.



3.4.11 `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`)

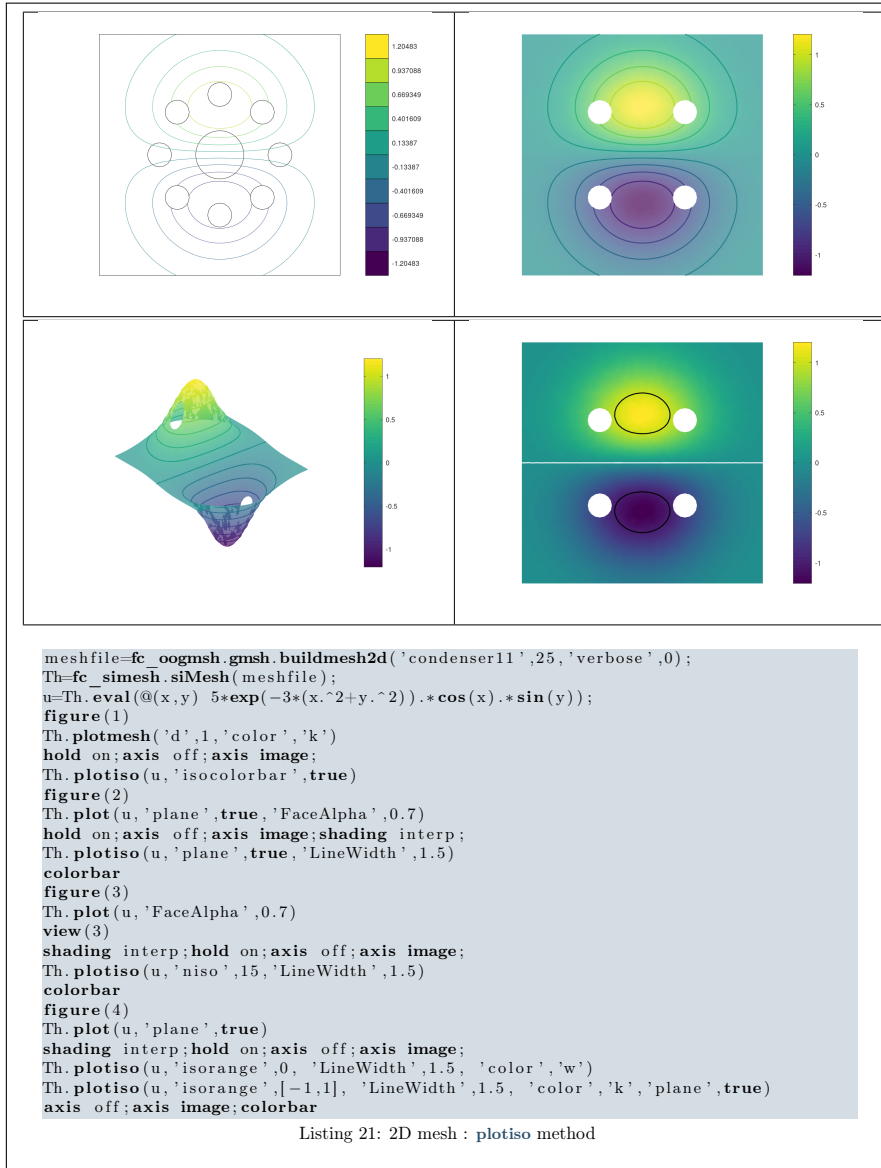
The options of second level are all options of

- `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`

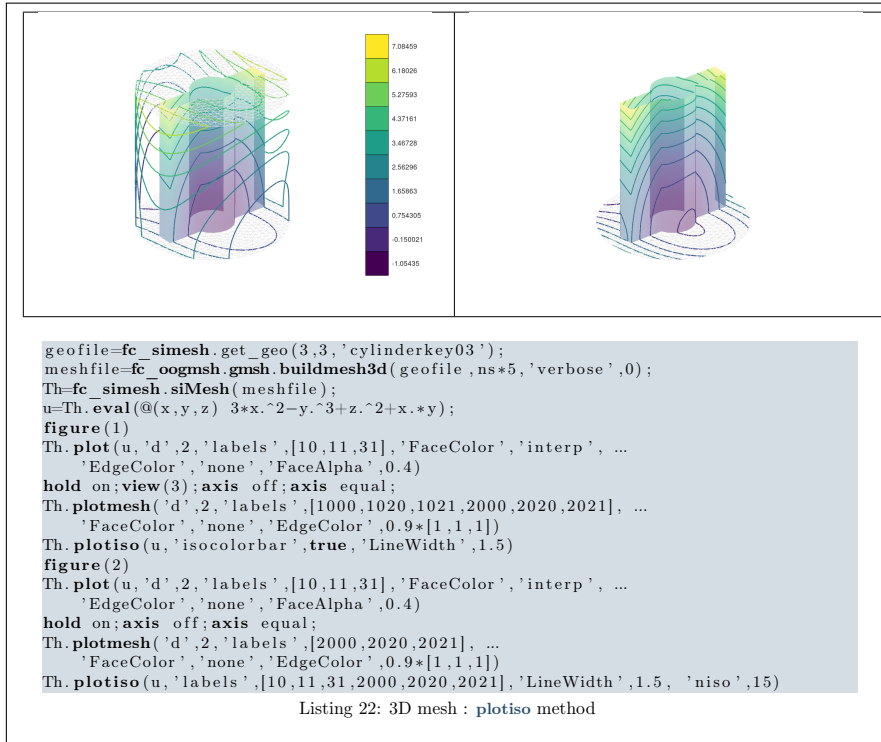
This function accepts until 4 output arguments :

- 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.

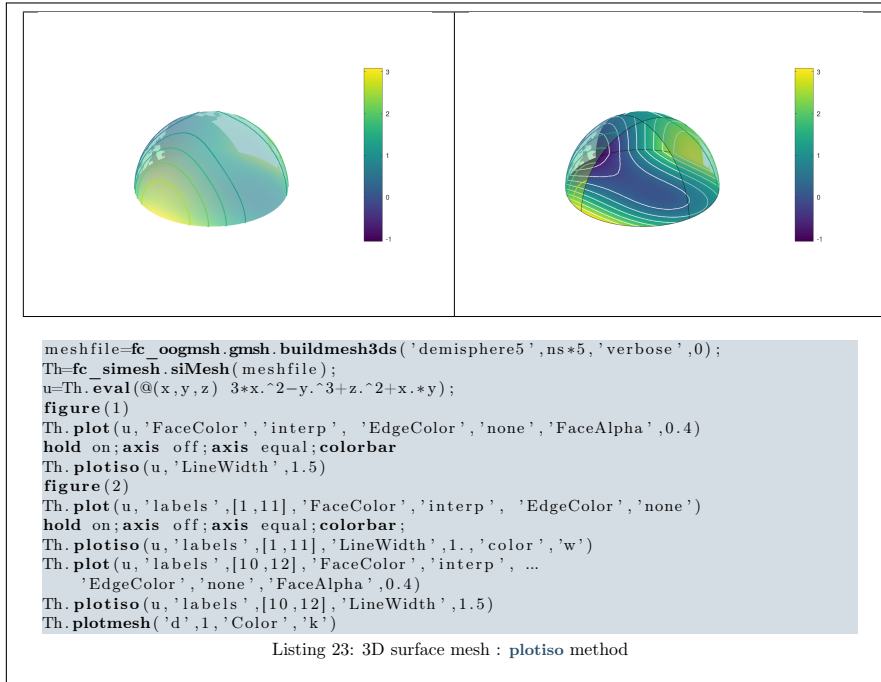
2D example The following example use the *.geo* file `condenser11.geo` which is in the directory `geodir` of the toolbox.



3D example The following example use the *.geo* file *cylinderkey.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.



3D surface example The following example use the `.geo` file `demisphere5.geo` which is in the directory `geodir` of the toolbox. This file contains description of a 3D surface mesh with simplices of dimensions 1 and 2.



3.4.12 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 , ...)
```

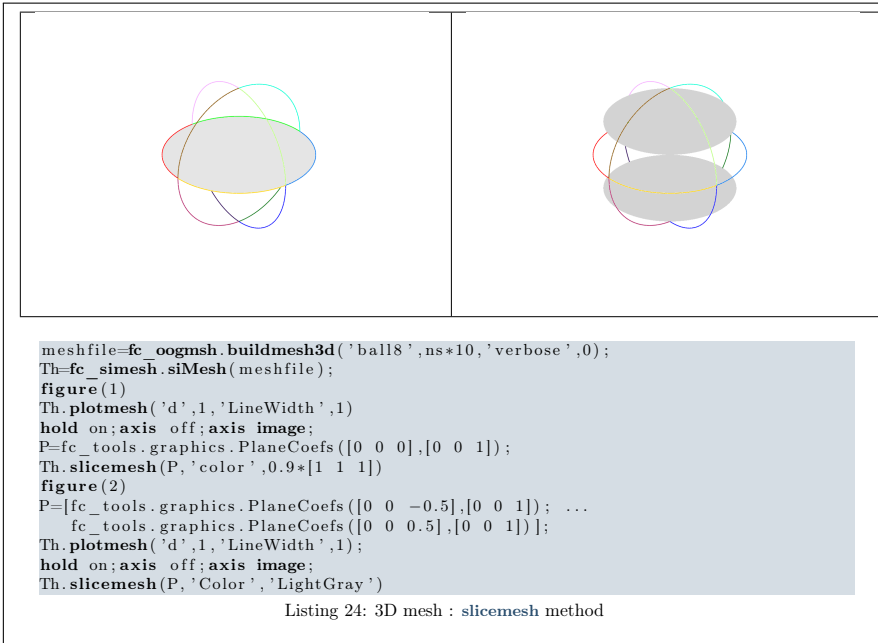
Description

`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 `fc_tools` 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.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.



3.4.13 slice method

The method `slice` displays 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.slice(u,P)
Th.slice(u,P,Name,Value, ...)
```

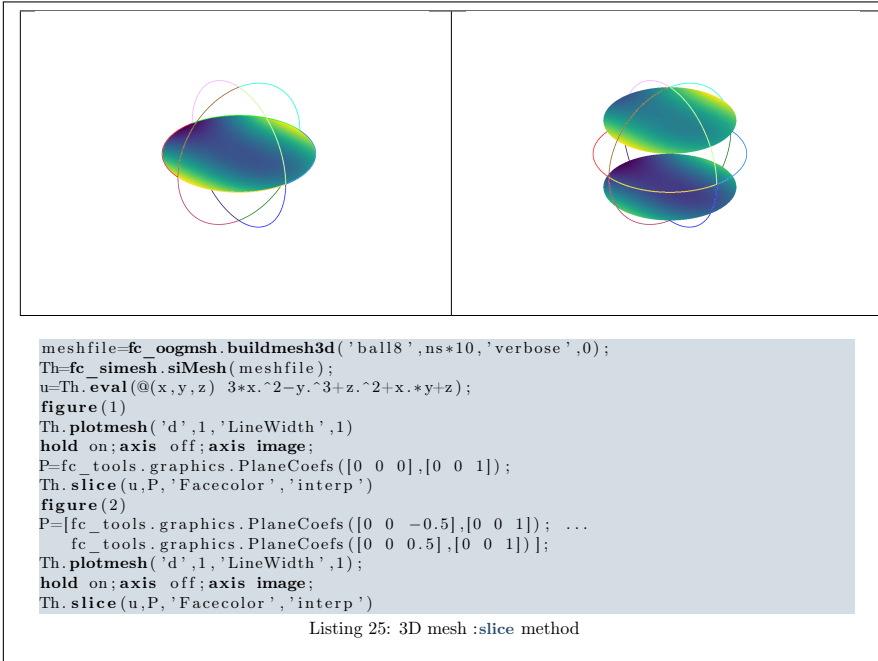
Description

`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 `fc_tools` 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.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.



3.4.14 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 `fc_tools` 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

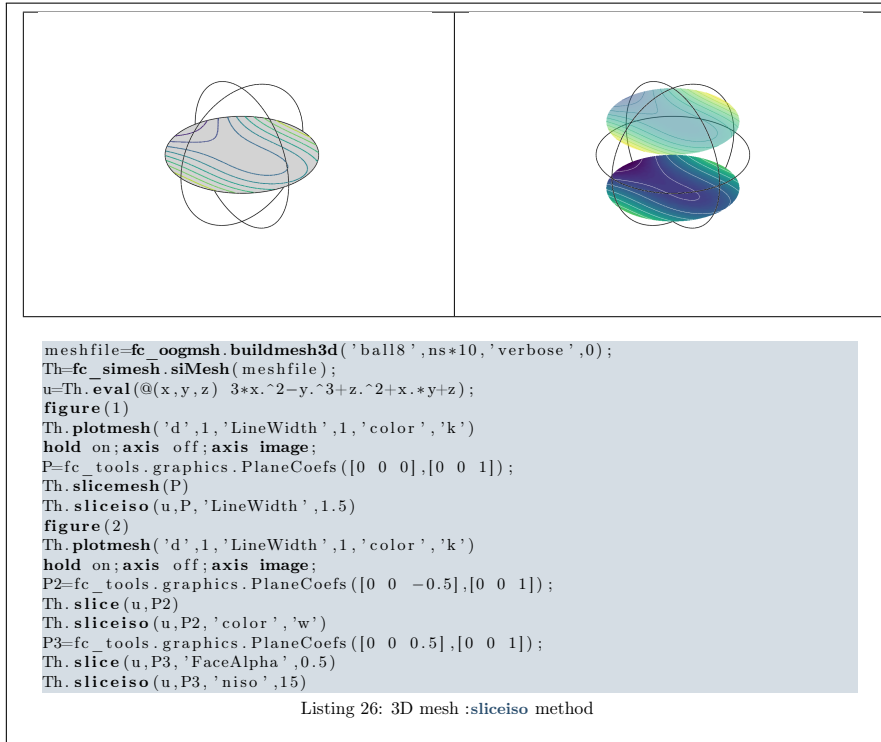
- `'labels'` : to select the labels of the elements to intersect,
- `'niso'` : to specify the number of isolines (default : 10)
- `'isorange'` : to specify the list of isovalues (default : empty)
- `'color'` : to specify one color for all isolines (default : empty)
- `'isocolorbar'` : if true display 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.



3.4.15 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- $Th.nq$.

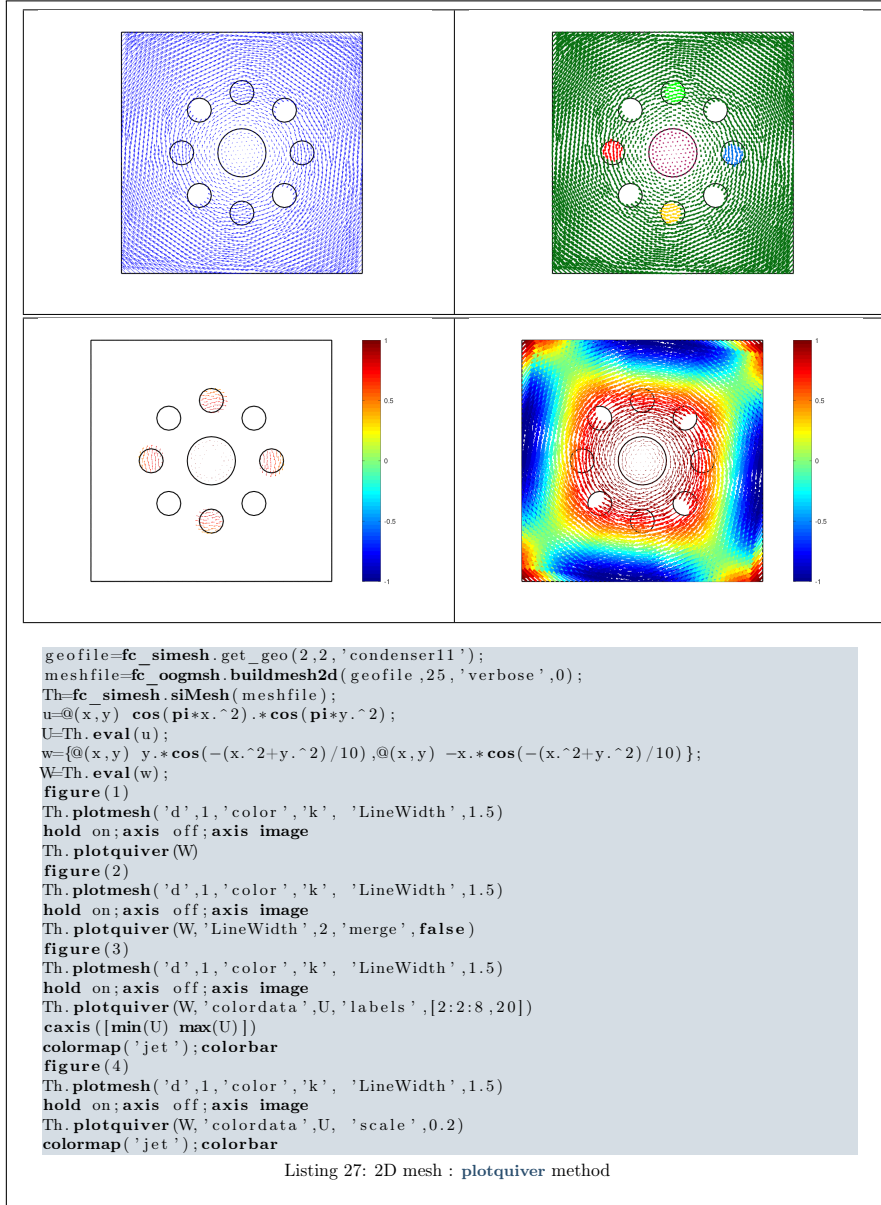
Th.plotquiver(V,Key,Value,...) specifies function options using one or more **Key,Value** pair arguments. Options of first level are

- **'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).

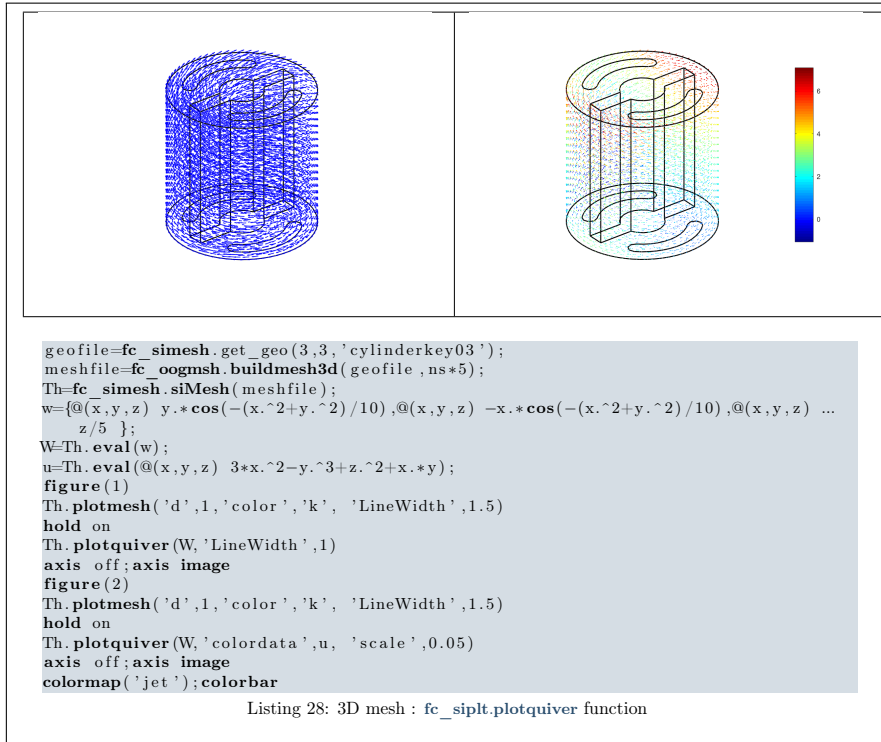
The options of second level depend on space dimension and **'colordata'** option. One can use any option of the following functions

- **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`.

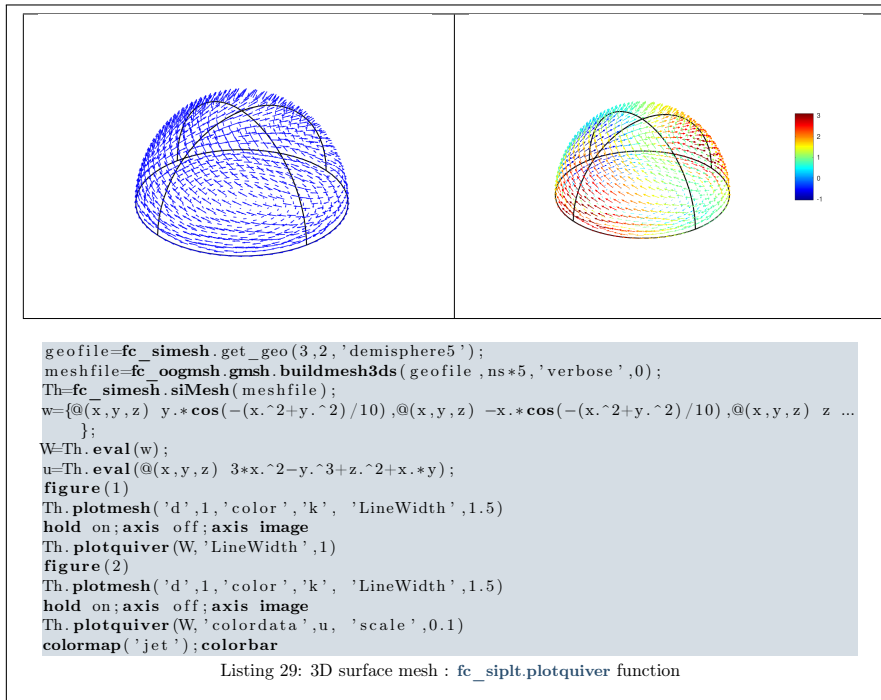
2D example The following example use the *.geo* file `condenser11.geo`.



3D example The following example use the *.geo* file `cylinderkey03.geo`. This file contains description of a 3D mesh with simplices of dimensions 1, 2 and 3.



3D surface example The following example use the `.geo` file `demisphere5.geo` which is in the directory `geodir` of the toolbox. This file contains description of a 3D surface mesh with simplices of dimensions 1 and 2.



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 30 a usage example generating a 2D hypercube as a `fc_simesh.siMesh` object is given. This `fc_simesh.siMesh` object is representing in Figure 4 by using the `FC-SIPLT` package.

Listing 30: 2D Hypercube `fc_simesh.siMesh` object generated with the function `siMesh.HyperCube`

```
Th=fc_simesh.hypercube(2,10);
disp(Th)
```

Output

```
fc_simesh.siMesh with properties:
  bbox: [ 0 1 0 1 ] (1x4 double)
    d: 2 double
  dim: 2 double
   nq: 121 double
nqParents: 121 double
  nsTh: 9 double
  other: []
   sTh: (1x9 cell)
sThboundlabs: []
 sThcolors: (9x3 double)
 sThgeolab: []
   sThlab: [ 1 1 2 3 4 1 2 3 4 ] (1x9 double)
sThpartlabs: []
 sThphyslab: 1 double
 sThsimp: [ 2 1 1 1 1 0 0 0 0 ] (1x9 double)
toGlobal: (1x121 double)
toParent: (1x121 double)
toParents: (1x1 cell)
```

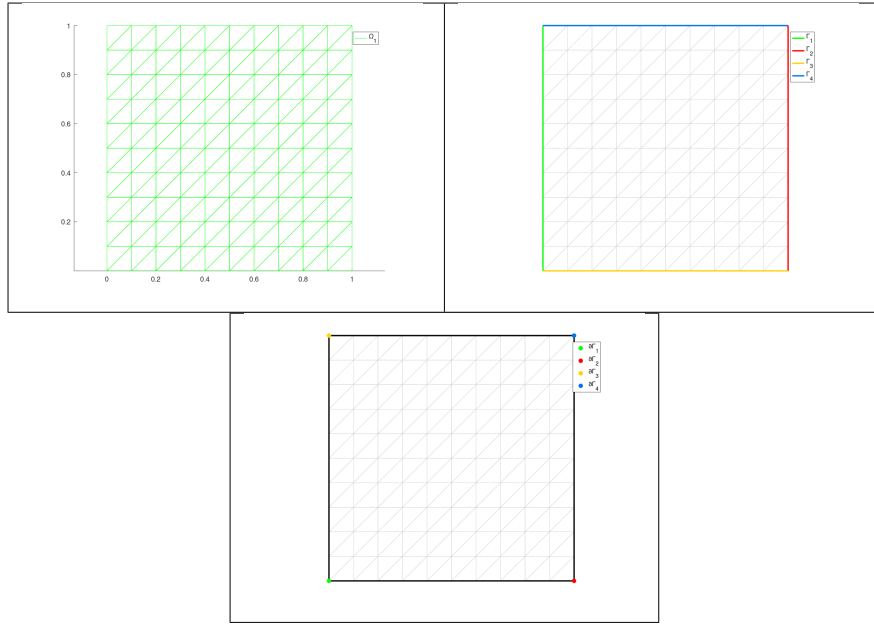


Figure 4: 2D Hypercube `fc_simesh.siMesh` object generated with the function `fc_simesh.hypercube`, representation of the elementary meshes with 2-simplices (top left), 1-simplices (top right) and 0-simplices (bottom)

3.5.2 3D hypercube

In Listing 31 a usage example generating a 3D hypercube as a `fc_simesh.siMesh` object is given. This `fc_simesh.siMesh` object is representing in Figure 5 by using the the `FC-SIPLT` package. .

Listing 31: 3D Hypercube `fc_simesh.siMesh` object generated with the function `fc_simesh.hypercube`

```
Th=fc_simesh.hypercube(3,10);
disp(Th)
```

Output

```
fc_simesh.siMesh with properties:
    bbox: [ 0 1 0 1 0 1 ] (1x6 double)
    d: 3 double
    dim: 3 double
    nq: 1331 double
    nqParents: 1331 double
    nsTh: 27 double
    other: []
    sTh: (1x27 cell)
    sThboundlabs: []
    sThcolors: (27x3 double)
    sThgeolab: []
    sThlab: (1x27 double)
    sThpartlabs: []
    sThphyslab: 1 double
    sThsimp: (1x27 double)
    toGlobal: (1x1331 double)
    toParent: (1x1331 double)
    toParents: (1x1 cell)
```

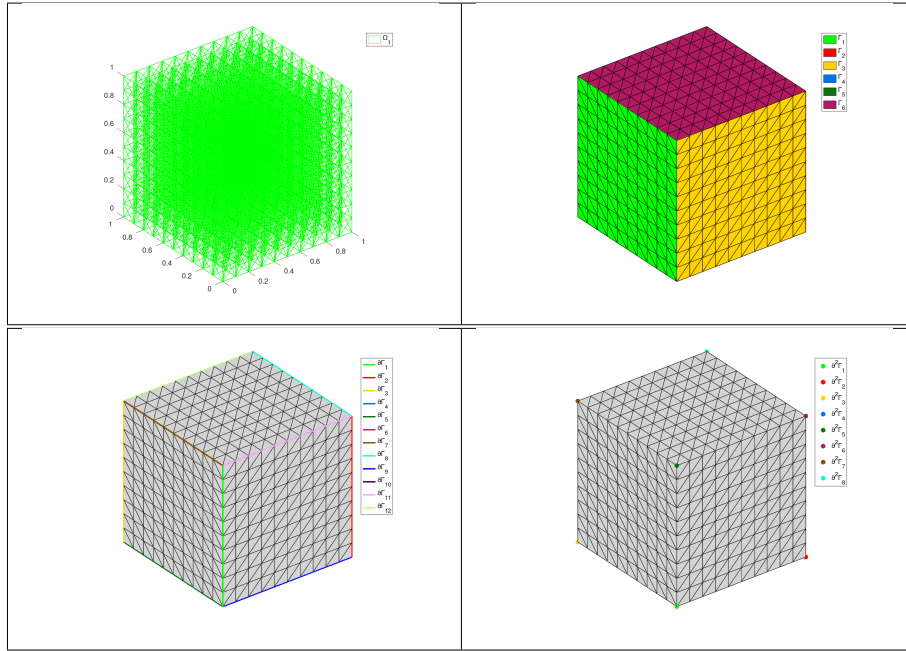
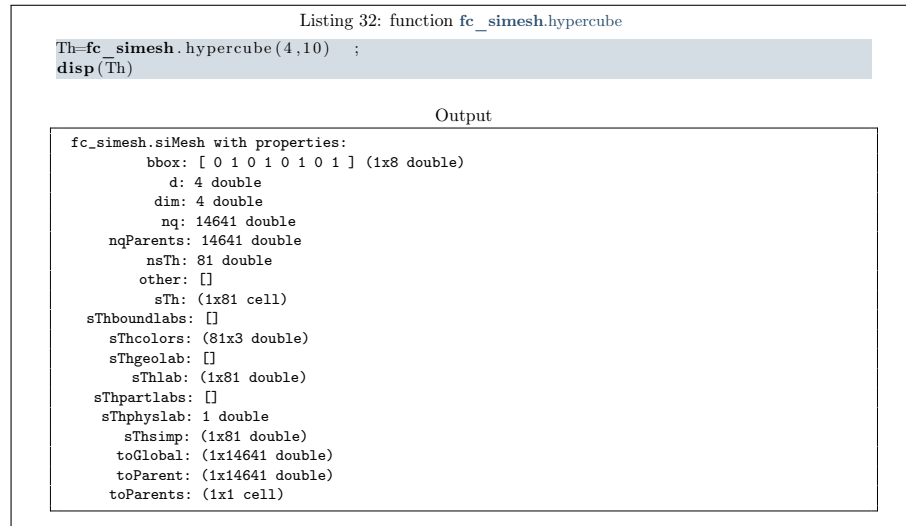


Figure 5: 3D Hypercube `fc_simesh.siMesh` object generated with the function `siMesh.HyperCube`, representation of the elementary meshes with 3-simplices (top left), 2-simplices (top right), 1-simplices (bottom left) and 0-simplices (bottom right)

3.5.3 4D hypercube

In Listing 32 a usage example generating a 4D hypercube as a `fc_simesh.siMesh` object is given.



3.5.4 5D hypercube

In Listing 32 a usage example generating a 5D hypercube as a `fc_simesh.siMesh` object is given.

Listing 33: 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)  
    sThboundlabs: []  
        sThcolors: (243x3 double)  
        sThgeolab: []  
        sThlab: (1x243 double)  
    sThpartlabs: []  
    sThphyslab: 1 double  
        sThsimp: (1x243 double)  
        toGlobal: (1x16807 double)  
        toParent: (1x16807 double)  
        toParents: (1x1 cell)
```

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>	9
3	3D Mesh from <code>quart_sphere2.geo</code>	11
4	3D surface Mesh from <code>demisphere4surf.geo</code>	13
5	<code>fc_simesh.siMesh</code> constructor	15
6	<code>fc_simesh.siMesh</code> find method samples	16
7	<code>feval</code> method, four ways to defined a function	17
8	<code>feval</code> method with a vector-valued function	17
9	<code>eval</code> method, four ways to defined a function	18
10	<code>eval</code> method with a vector-valued function	18
11	<code>get_mesh</code> method, four ways to defined a function	19
12	<code>get_nme</code> method	20
13	<code>get_nqe</code> method	20
14	2D mesh : <code>plotmesh</code> method	22
15	3D plot mesh	23
16	3D mesh : <code>plotmesh</code> method	24
17	3D surface mesh : <code>plotmesh</code> method	25
18	2D mesh : <code>plot</code> method	27
19	3D mesh : <code>plot</code> method	28
20	3D surface mesh : <code>plot</code> method	29
21	2D mesh : <code>plotiso</code> method	31
22	3D mesh : <code>plotiso</code> method	32
23	3D surface mesh : <code>plotiso</code> method	32
24	3D mesh : <code>slicemesh</code> method	33
25	3D mesh : <code>slice</code> method	34
26	3D mesh : <code>sliceiso</code> method	36
27	2D mesh : <code>plotquiver</code> method	37
28	3D mesh : <code>fc_siplt.plotquiver</code> function	38
29	3D surface mesh : <code>fc_siplt.plotquiver</code> function	38
30	2D Hypercube <code>fc_simesh.siMesh</code> object generated with the function <code>siMesh.HyperCube</code>	39
31	3D Hypercube <code>fc_simesh.siMesh</code> object generated with the function <code>fc_simesh.hypercube</code>	40
32	function <code>fc_simesh.hypercube</code>	41
33	function <code>siMesh.HyperCube</code>	42

B References

- [1] F. Cuvelier. `fc_oogmsh`: an object-oriented Octabe 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 : 0.4.1
commit : 6a226cf06d42c576ab67fe23a2cb0d5e700effae
date : 2020-03-08
time : 07-13-43
status : 0
-----
name : fc-tools
tag : 0.0.31
commit : 3a25f08630ff69e7f8ef4ab451671e454c492e36
date : 2020-02-23
time : 11-27-44
status : 0
-----
name : fc-bench
tag : 0.1.2
commit : 666dc60d1277f5fa9c99dee4ae1c33270f22c57d
date : 2020-02-16
time : 06-38-46
status : 0
-----
name : fc-hypermesh
tag : 1.0.3
commit : c520b34cfd7eb0dbf9e4ecd459fd7162db73cc58
date : 2020-02-16
time : 08-34-19
status : 0
-----
name : fc-amat
tag : 0.1.2
commit : 957340f6e71d805dbd8b9d04c434b24fd3f92591
date : 2020-02-16
time : 06-39-42
status : 0
-----
name : fc-meshtools
tag : 0.1.3
commit : cdbc41bc98af4e4facc1746024aced1f21aae53
date : 2020-02-17
time : 10-52-56
status : 0
-----
name : fc-graphics4mesh
tag : 0.1.2
commit : e922041e090275b69207ed6d4fb52df06f5e3fba
date : 2020-03-07
time : 16-09-48
status : 0
-----
name : fc-oogmsh
tag : 0.2.3
commit : 614c8154b265a17a6dd31e328100974ea7e9e0bc
date : 2020-03-06
time : 05-19-37
status : 0
-----
name : fc-siplt
tag : 0.2.1
commit : 1ff1ea26e27f75eabba537e9c9e5dd6cc2346684
date : 2020-03-07
time : 15-58-27
status : 0
-----

```

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

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

Using the remote configuration repository:

```
url      ssh://lagagit/MCS/Cuvelier/Matlab/fc-config  
commit   81d0561dcc0535351928c138132cebb80deef3d
```