



FC_SIMESH_MATPLOTLIB package, User's Guide *

François Cuvelier[†]

May 2, 2017

Abstract

The FC_SIMESH Python package allows to use simplices meshes generated from `gmsh` (in dimension 2 or 3) or an hypercube triangulation (in any dimension). The FC_SIMESH_MAYAVI Python package presented in this report is an add-on to the FC_SIMESH Python package. A particular care was taken to the graphics representations of meshes and datas on meshes by using `mayavi` package.

Contents

1	Functions of the FC_SIMESH_MATPLOTLIB package	2
1.1	function <code>PLOTMESH</code>	2
1.1.1	2D example	3
1.1.2	3D example	4
1.1.3	3D surface example	6
1.2	function <code>PLOT</code>	7
1.2.1	2D example	8
1.2.2	3D example	9
1.2.3	3D surface example	10
1.3	function <code>PLOTISO</code>	11
1.3.1	2D example	11

*Compiled with Python 3.6.0, packages FC_HYPERMESH-0.0.4, FC_OOGMSH-0.0.3, FC_TOOLS-0.0.9, FC_SIMESH-0.0.4 and the plotting libraries MAYAVI-4.5.0, FC_SIMESH_MAYAVI-4.5.0

[†]Université Paris 13, Sorbonne Paris Cité, LAGA, CNRS UMR 7539, 99 Avenue J-B Clément, F-93430 Villetaneuse, France, cuvelier@math.univ-paris13.fr.

This work was partially supported by ANR Dedales.

1.3.2	3D example	12
1.3.3	3D surface example	13
1.3.4	function QUIVER	14
1.3.5	2D example	15
1.3.6	3D example	15
1.3.7	3D surface example	16
1.4	function SLICEMESH	17
1.5	function SLICE	19
1.5.1	3D example	20
1.6	function SLICEISO	21
1.6.1	3D example	22
1.7	function STREAMLINE	23

1 Functions of the `fc_simesh_matplotlib` package

1.1 function **PLOTMESH**

The **PLOTMESH** function displays the mesh or parts of the mesh defined by an **siMESH** object.

Syntaxe

```
simlab.plotmesh(Th, )
simlab.plotmesh(Th, Key=Value, ...)
```

Description

`simlab.plotmesh(Th,)` displays all the `Th.d`-dimensional simplices elements.

`simlab.plotmesh(Th,Key=Value, ...)` specifies function options using one or more `Key,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),
- `legend` : add a legend to graph if `True` (default : `False`)
- `cute_planes'` : cut mesh by `n` plans given by a list of `Plane` objects (only in dimension 3). The `Plane` constructor is `Plane(origin=[x,y,z], normal=[nx,ny,z])` which defined the plane coming through point `origin` and orthogonal to the vector `normal`. The normal vector pointed to the part of the mesh not displayed. (only in dimension 3) default : `[]` (no cut).

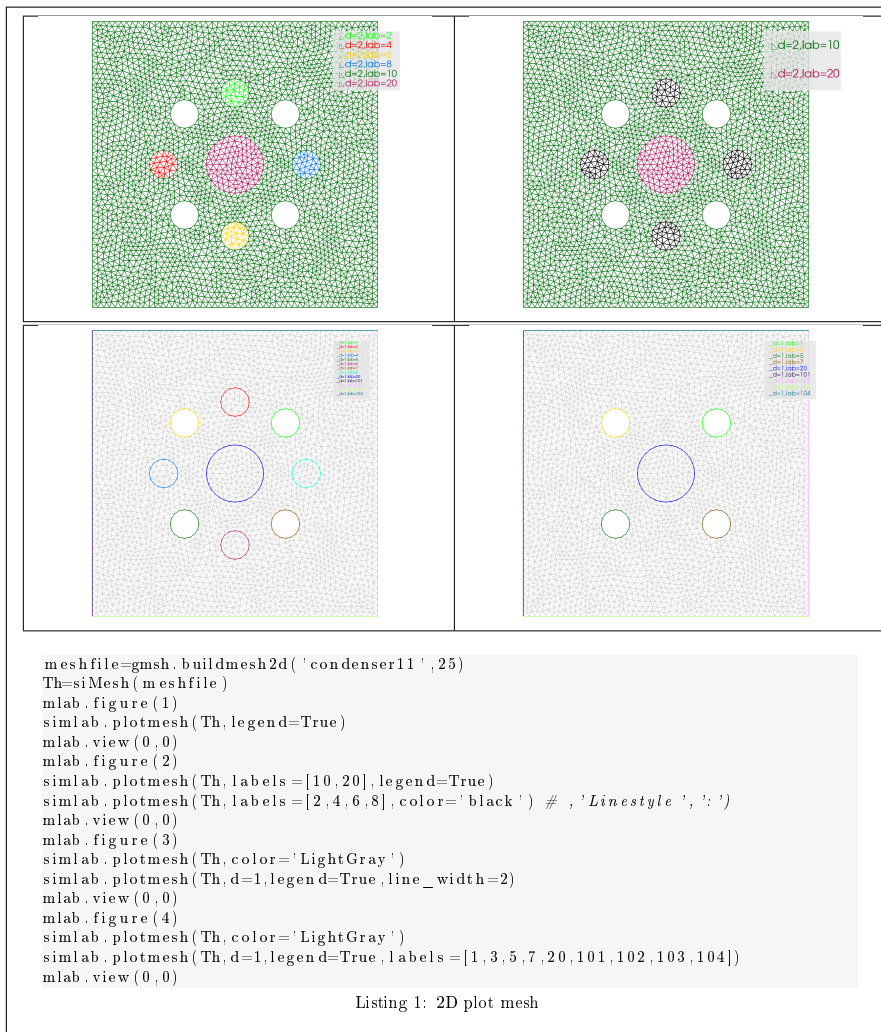
The options of second level depend on the type of elementaries mesh elements to represent.

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

- In dimension 3,
 - if $d == 3$, `mlab.pipeline.surface` function is used with `vtk.UnstructuredGrid` and `vtk.Tetra().cell_type`
 - if $d == 2$, `mlab.pipeline.surface` function is used with `vtk.UnstructuredGrid` and `vtk.Triangle().cell_type`
 - if $d == 1$, `mlab.pipeline.surface` function is used with `vtk.UnstructuredGrid` and `vtk.Line().cell_type`
 - if $d == 0$, not yet implemented
- In dimension 2,
 - if $d == 2$, `mlab.triangular_mesh` function is used,
 - if $d == 1$, if $d == 1$, `mlab.pipeline.surface` function is used with `vtk.UnstructuredGrid` and `vtk.Line().cell_type`
 - if $d == 0$, not yet implemented
- dimension 1, not yet implemented

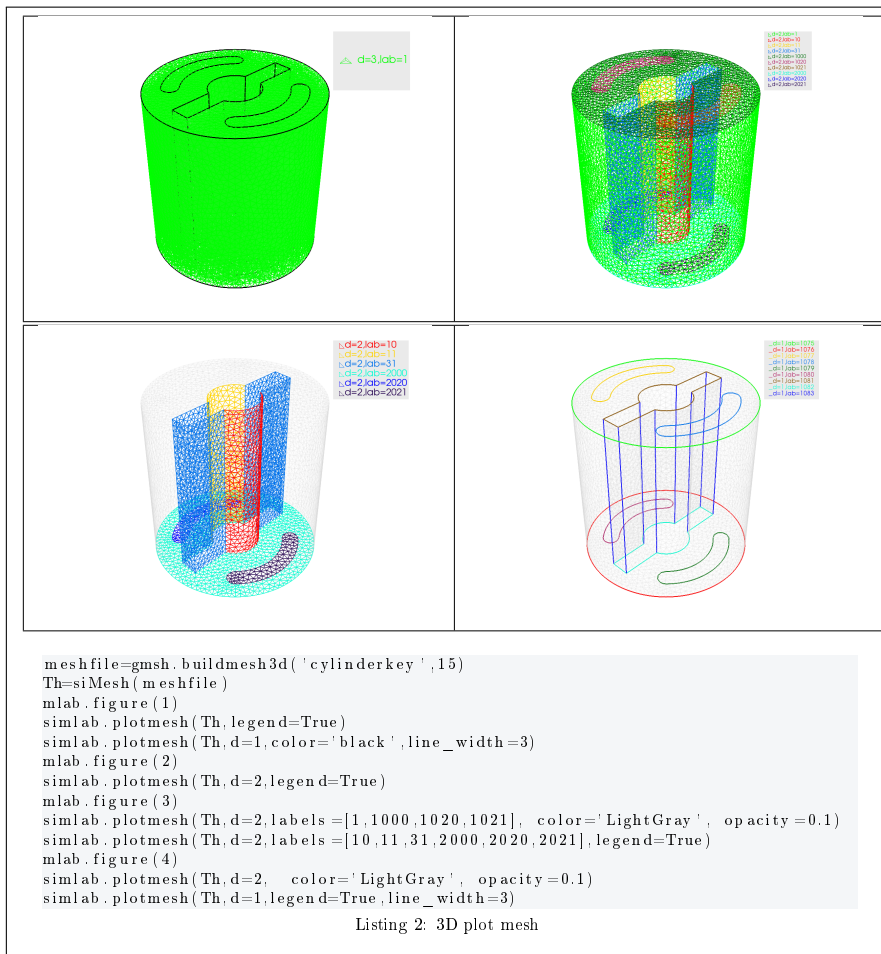
1.1.1 2D example

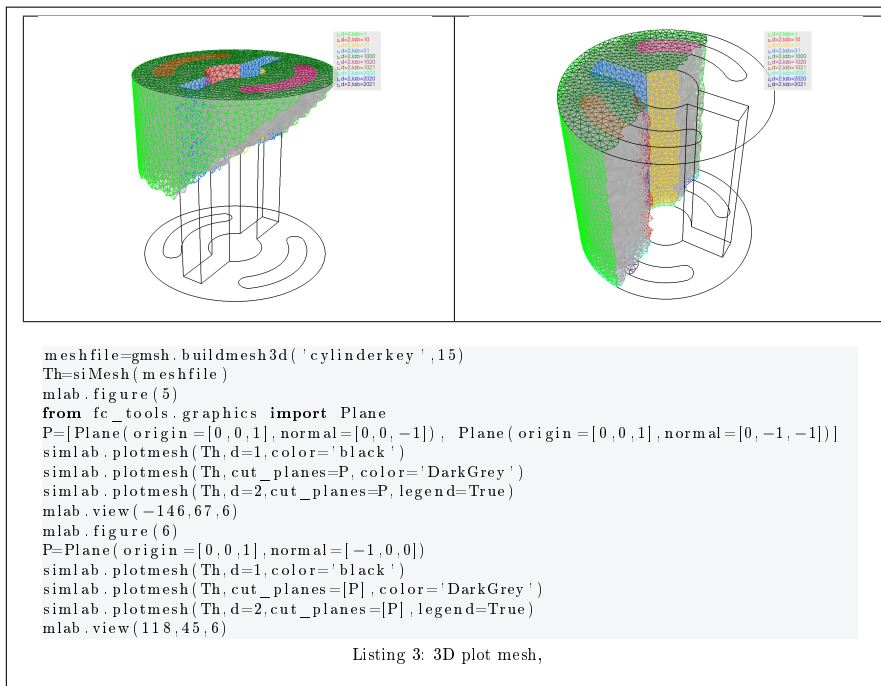
The following example use the `.geo` file `condenser11.geo` which is in the directory `geodir` of the toolbox



1.1.2 3D example

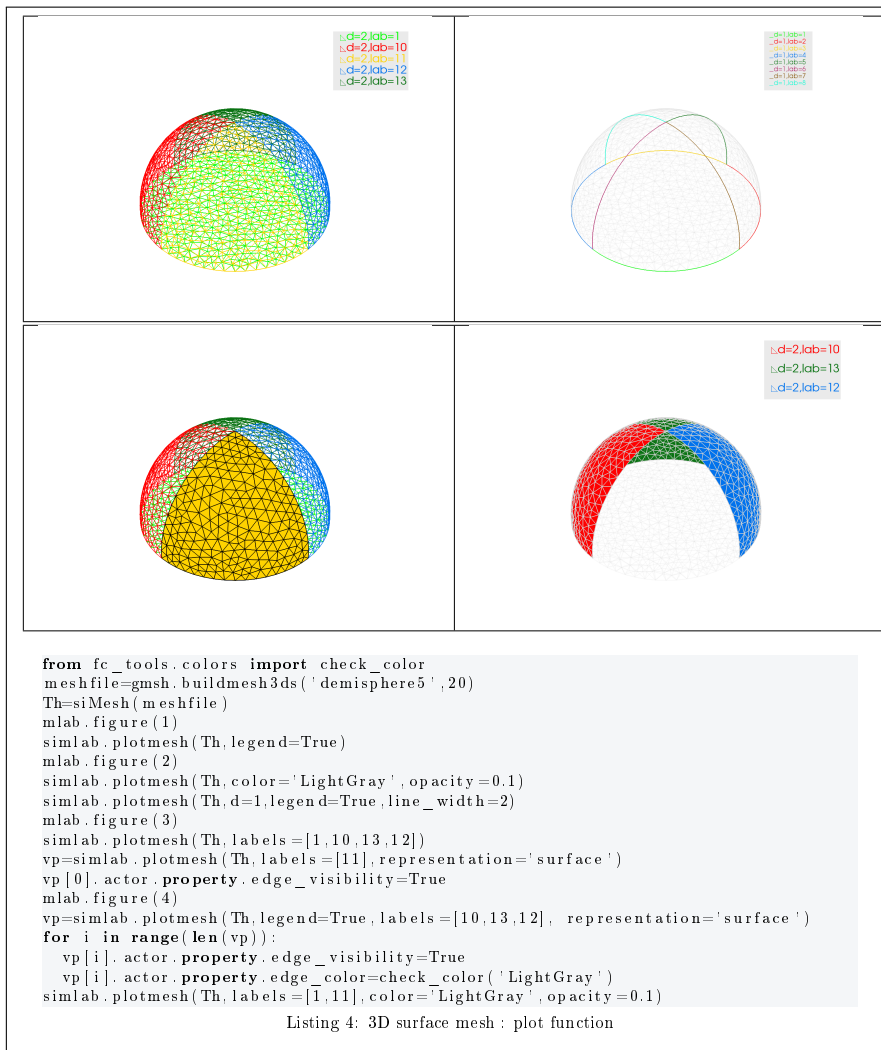
The following example use the *.geo* file *cylinderkey.geo* which is in the directory *geodir* of the toolbox. This file contains description of a 3D mesh with simplices of dimensions 1, 2 and 3.





1.1.3 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.



1.2 function PLOT

The **PLOT** function displays scalar data on the mesh or parts of the mesh defined by an **siMESH** object.

Syntaxe

```

simlab.plot(Th,u)
simlab.plot(Th,u,Key=Value,...)

```

Description

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

`simlab.plot(Th,u,Key=Value, ...)` specifies function options using one or more Key,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, (default : False)

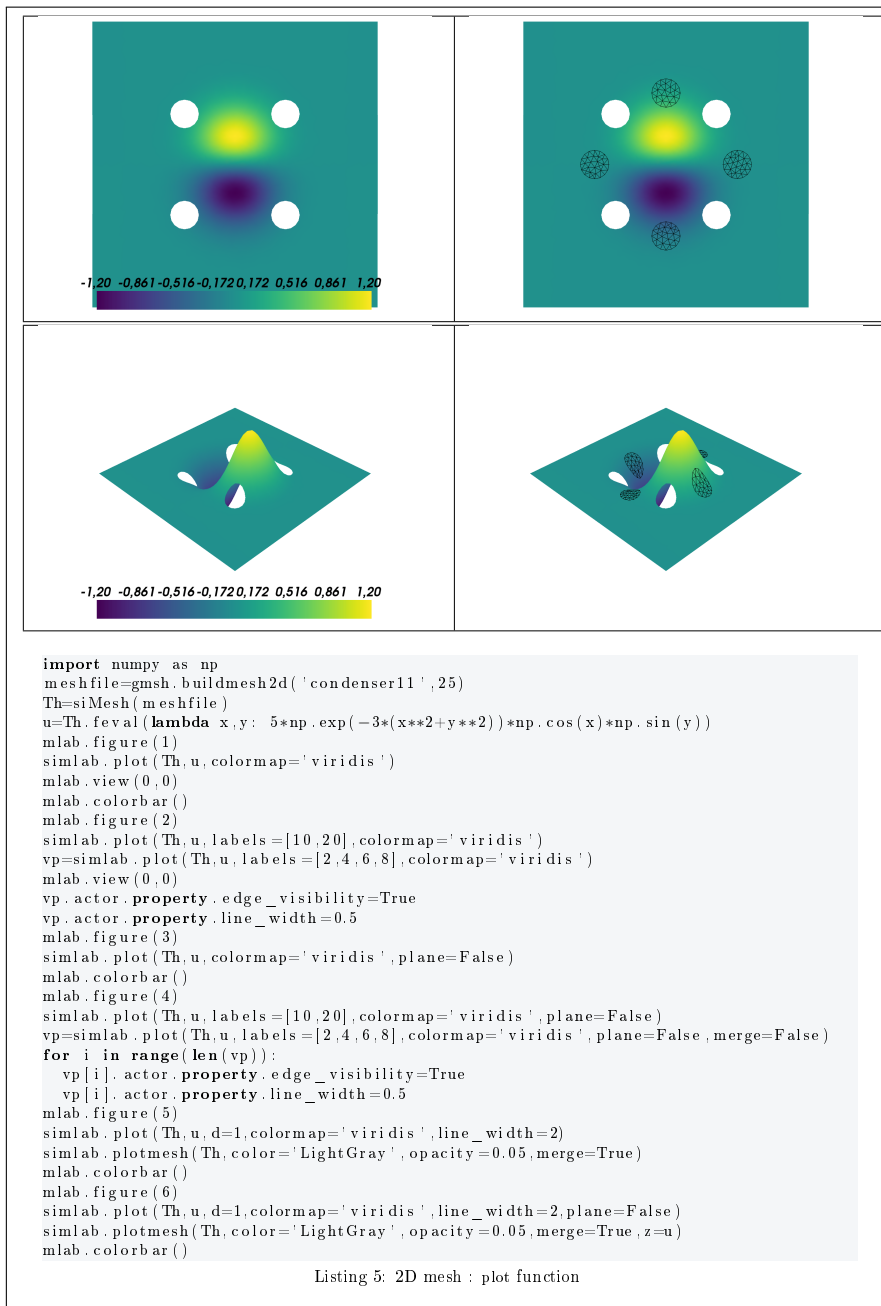
The second level options depend on the type of elementaries mesh elements on which we want to represent datas.

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

- In dimension 3,
 - if $d == 3$, `mlab.pipeline.surface` function is used with `vtk.UnstructuredGrid` and `vtk.Tetra().cell_type`
 - if $d == 2$, `mlab.triangular_mesh` function is used.
 - if $d == 1$, `mlab.pipeline.surface` function is used with `vtk.UnstructuredGrid` and `vtk.Line().cell_type`
- In dimension 2,
 - if $d == 2$, `mlab.triangular_mesh` function is used.
 - if $d == 1$, `mlab.pipeline.surface` function is used with `vtk.UnstructuredGrid` and `vtk.Line().cell_type`.
- Dimension 1 : not implemented.

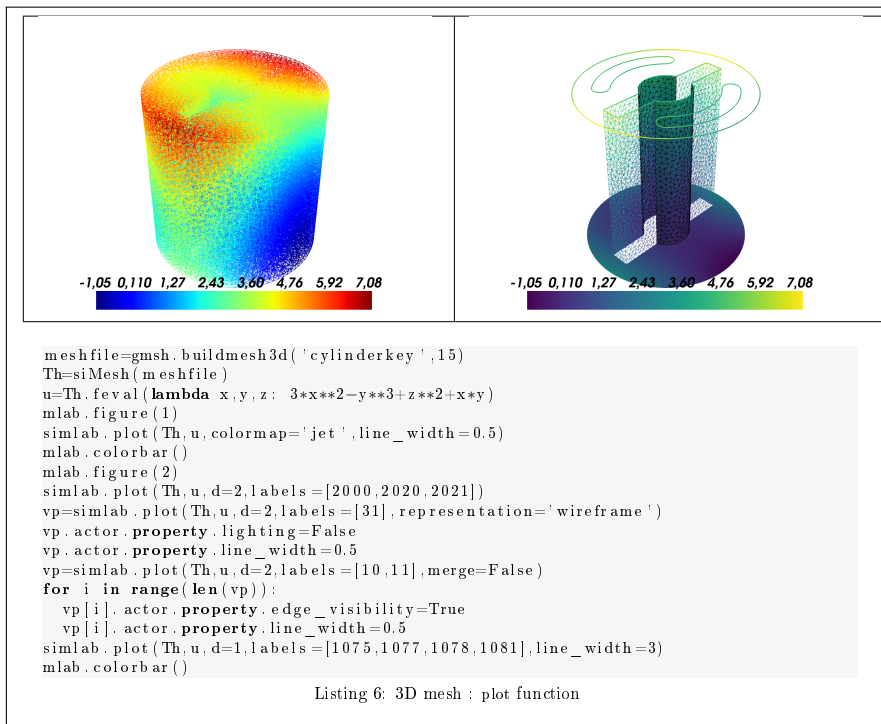
1.2.1 2D example

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



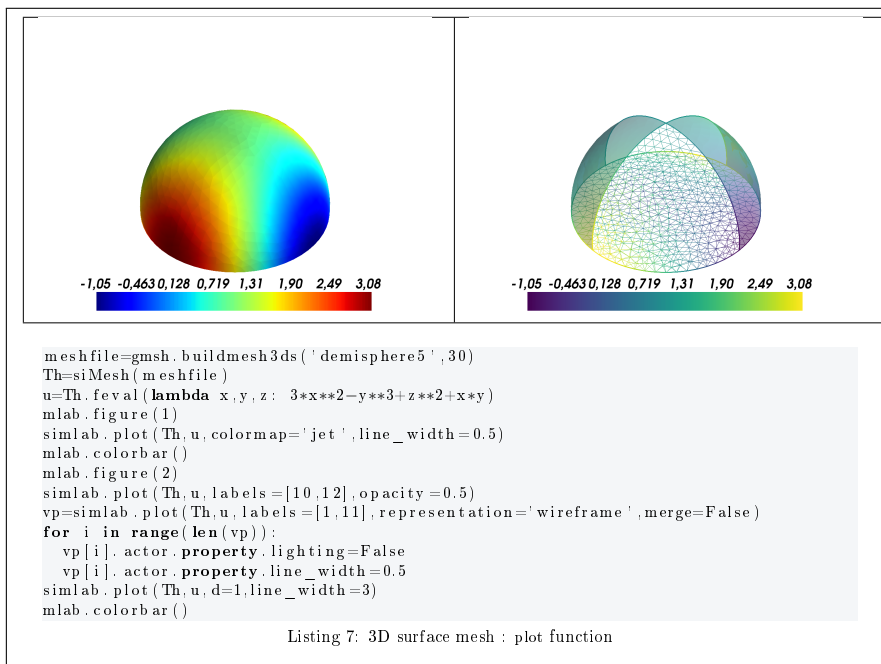
1.2.2 3D example

The following example uses 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.



1.2.3 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.



1.3 function `PLOTISO`

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

Syntaxe

```
simlab.plotiso(Th,u)
simlab.plotiso(Th,u,Key=Value, ...)
```

Description

`simlab.plotiso(Th,u)` displays data `u` on all the 2-dimensional simplices elements as colored isovalues. The data `u` is an 1D-array of size `Th.nq` or `Th.nqGlobal` or `Th.nqParent`.

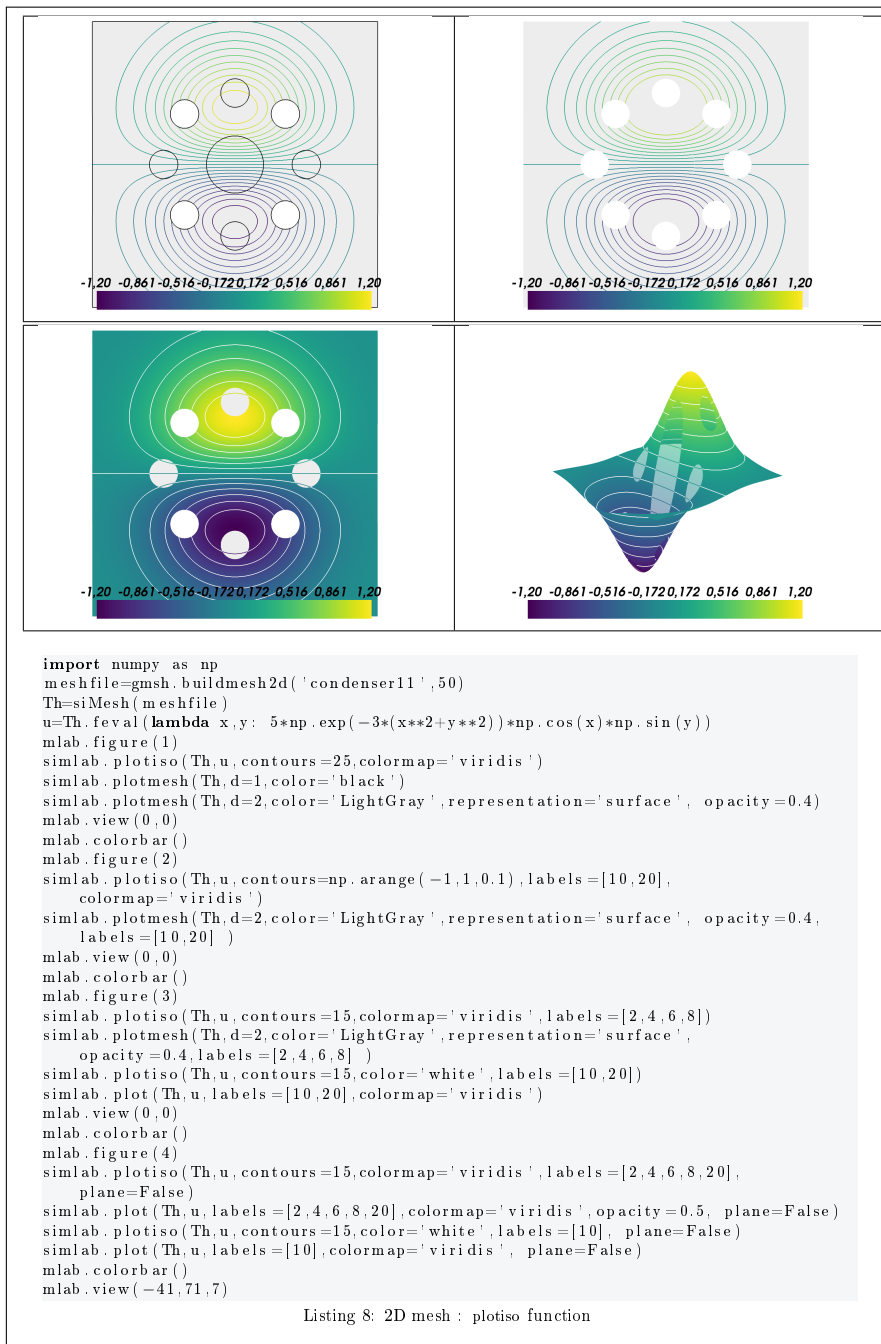
`simlab.plotiso(Th,u,key=value, ...)` specifies function options using one or more key,value pair arguments. Options of first level are

- `contours` : to specify the number of isolines (default : 10) or a list/numpy array of isovalues (default : empty)
- `labels` : to select the labels of the elements to display data,
- `plane` : if False draw 3D isovalues with data `u` as `z` values. (default : True)
- `color` : to specify one color for all isolines (default : None)

The second level options are the options of the `mlab.pipeline.iso_surface` which we use to draw the isovalues.

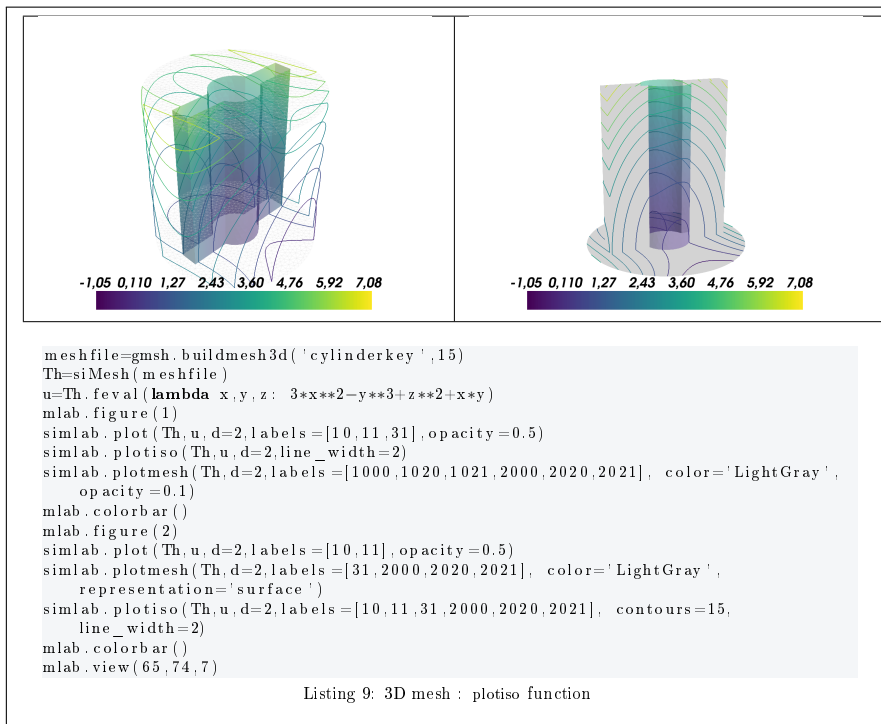
1.3.1 2D example

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



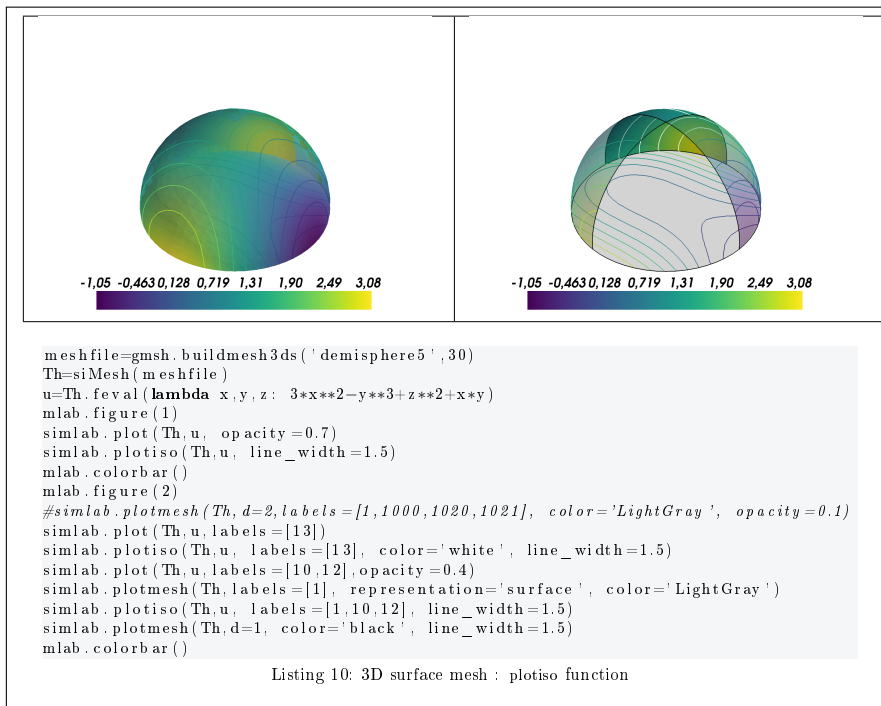
1.3.2 3D example

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.



1.3.3 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.



1.3.4 function **QUIVER**

The **QUIVER** function displays vector field datas on the mesh or parts of the mesh defined by an **siMESH** object.

Syntaxe

```

simlab.quiver(Th,V)
simlab.quiver(Th,V,Key=Value,...)

```

Description

`simlab.quiver(Th,V)` displays vector field V on each vertices of the d -dimensional simplices elements in dimension $d = 2$ or $d = 3$. The data V is an 2D-array numpy array of size dim -by- $Th.nq$.

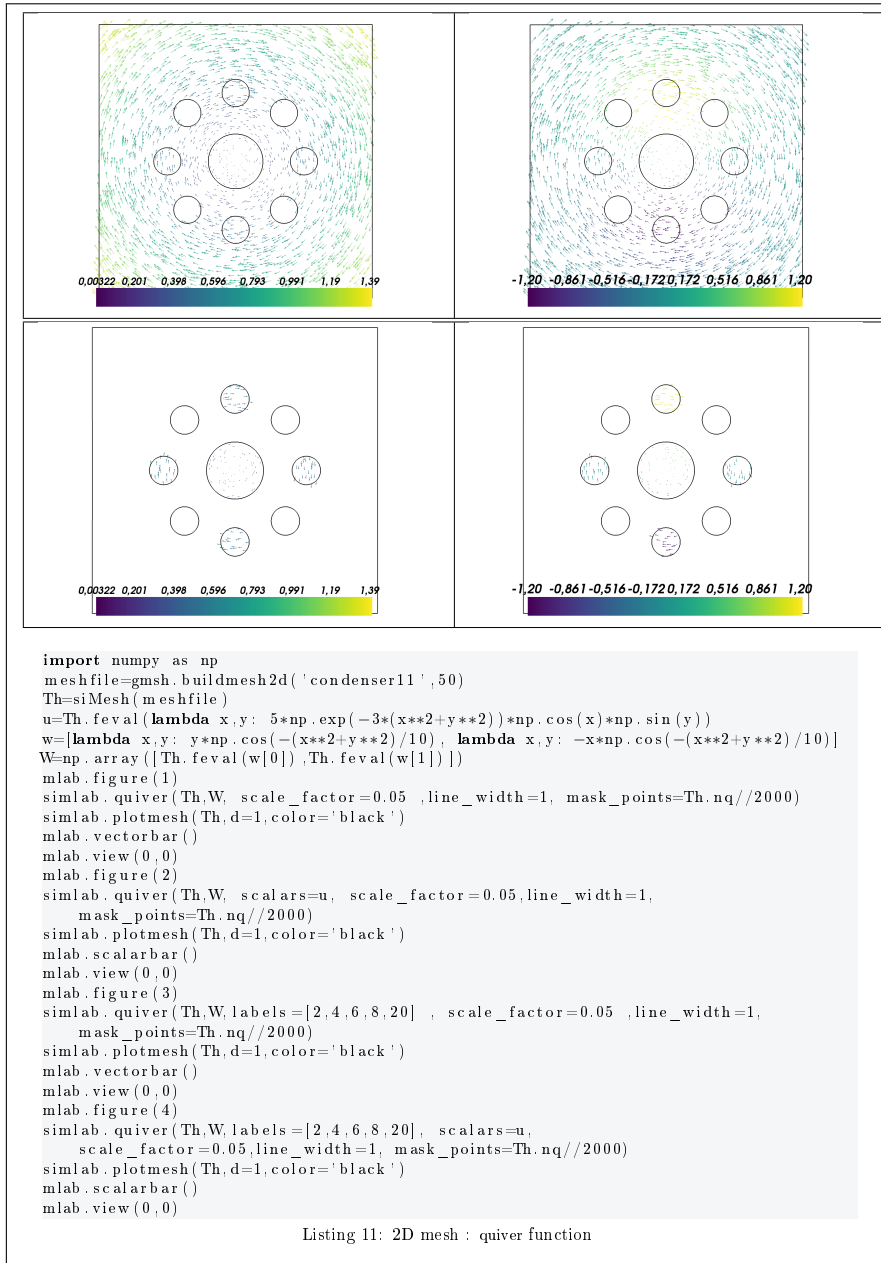
`simlab.quiver(Th,V,Key=Value,...)` specifies function options using one or more Key,Value pair arguments. Options of first level are

- **labels** : to select the labels of the elements to display data,
- **scalars** : to set quivers color to a numpy array of size $Th.nq$ (default : empty and use colors of the mesh elements).
- **color** : to specify one color for all quivers.

For key/value pairs, one could also used those of the `mlab.quiver3d` function.

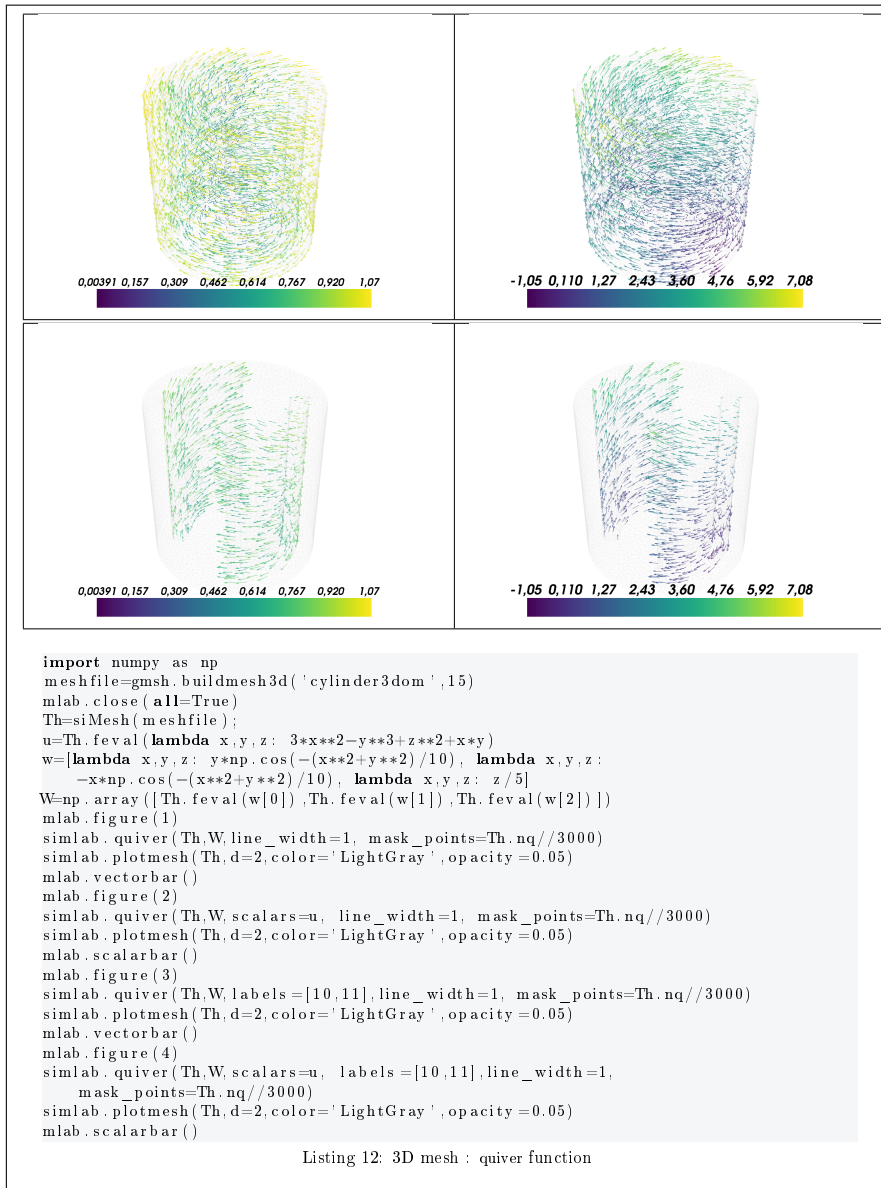
1.3.5 2D example

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



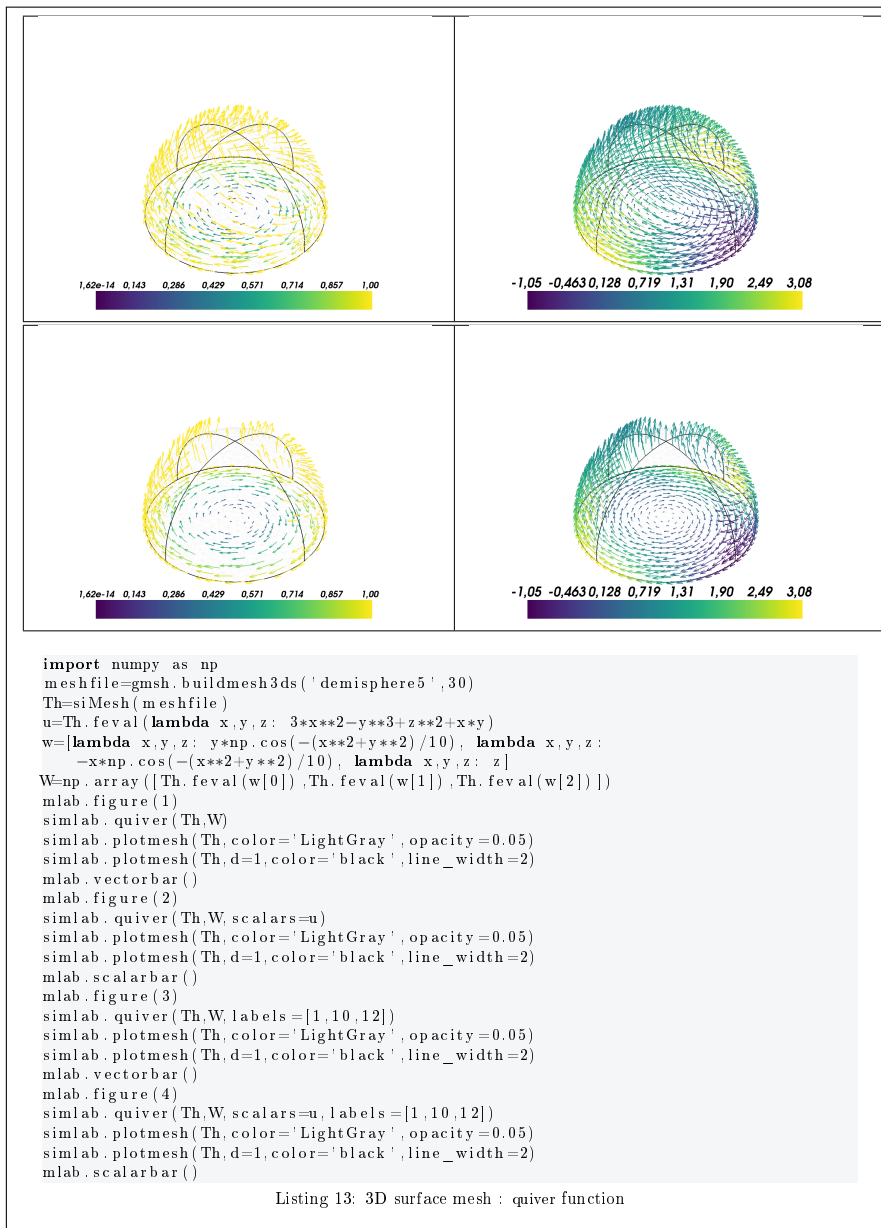
1.3.6 3D example

The following example use the `.geo` file `cylinderkey03.geo` which is in the directory `geodir/3d` of the toolbox. This file contains description of a 3D mesh with simplices of dimensions 1, 2 and 3.



1.3.7 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.



1.4 function SLICEMESH

The **SLICEMESH** function displays intersection of a plane and a 3D mesh or parts of a 3D mesh defined by an **siMESH** object.

Syntaxe

```

simlab.slicemesh(Th,)
simlab.slicemesh(Th,Key=Value,...)

```

Description


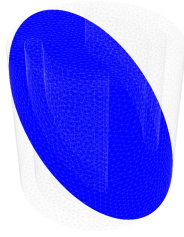
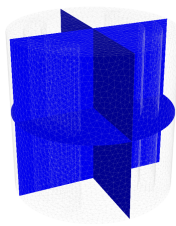
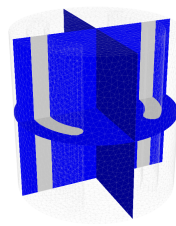
`simlab.slicemesh(Th,)` displays intersection of a plane and all the 3-dimensional simplices elements. By default the plane is given by an *origin* point $(0, 0, 0)$ which lies on it and a *normal* vector $(0, 0, 1)$ which is orthogonal to the plane.

`simlab.slicemesh(Th,Key=Value, ...)` specifies function options using one or more Key,Value pair arguments. Key could be:

- **origin** : to specify a point lying on the plane (default is $(0, 0, 0)$)
- **normal** : to specify a vector orthogonal to the plane (default is $(0, 0, 1)$)
- **labels** : to select the labels of the elements to intersect,

The other Key/Value pair options are those of `scalar_cut_plane` function from `mayavi.mlab.pipeline`.

The following example use the `.geo` file `cylinder3dom.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.

```

meshfile=gmsl.buildmesh3d('cylinder3dom',15);
mlab.close(all=True)
Th=siMesh(meshfile)
mlab.figure(1)
simlab.plotmesh(Th,legend=True)
simlab.plotmesh(Th,d=1,color='black',line_width=3)
mlab.figure(2)
simlab.plotmesh(Th,d=2,color='LightGray',opacity=0.05)
simlab.slicemesh(Th,origin=(0,0,1),normal=(-1,0,1))
mlab.view(132,53,7)
mlab.figure(3)
simlab.plotmesh(Th,d=2,color='LightGray',opacity=0.05)
for normal in [(1,0,0),(0,1,0),(0,0,1)]:
    simlab.slicemesh(Th,origin=(0,0,1),normal=normal)
mlab.view(155,66,7)
mlab.figure(4)
simlab.plotmesh(Th,d=2,color='LightGray',opacity=0.05)
for normal in [(1,0,0),(0,1,0),(0,0,1)]:
    simlab.slicemesh(Th,labels=[10,11],origin=(0,0,1),normal=normal,
        color='LightGray')
    simlab.slicemesh(Th,labels=[1],origin=(0,0,1),normal=normal)
mlab.view(155,66,7)

```

Listing 14: 3D mesh : slicemesh function

1.5 function SLICE

The **SLICE** function displays datas on the intersection of a plane and a 3D mesh or parts of a 3D mesh defined by an **siMESH** object.

Syntaxe

```
simlab.slice(Th,u)
simlab.slice(Th,u,Key=Value,...)
```

Description

`simlab.slice(Th,u)` displays `u` data on the intersection of a plane and all the 3-dimensional simplices elements. By default the plane is given by an *origin* point $(0,0,0)$ which lies on it and a *normal* vector $(0,0,1)$ which is orthogonal to the plane. The data `u` is an 1D-array of size `Th.nq` or `Th.nqGlobal` or `Th.nqParent`.

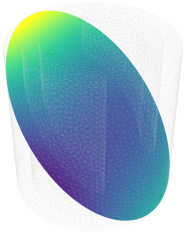
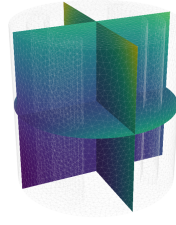
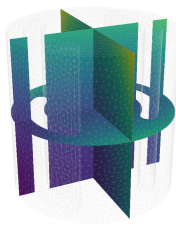
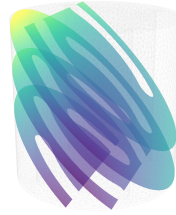
`simlab.slice(Th,u,Key,Value, ...)` specifies function options using one or more `Key,Value` pair arguments. Key could be:

- `origin` : to specify a point lying on the plane (default is $(0,0,0)$)
- `normal` : to specify a vector orthogonal to the plane (default is $(0,0,1)$)
- `labels` : to select the labels of the elements to intersect,

The other `Key/Value` pair options are those of `scalar_cut_plane` function from `mayavi.mlab.pipeline`.

1.5.1 3D example

The following example use the *.geo* file `cylinder3dom.geo` which is in the directory `geodir/3d` of the toolbox. This file contains description of a 3D mesh with simplices of dimensions 1, 2 and 3.

```

meshfile=gmesh.buildmesh3d('cylinder3dom',15)
mlab.close(all=True)
Th=siMesh(meshfile)
u=Th.feval(lambda x,y,z: 3*x**2-y**3+z**2+x*y)
mlab.figure(1)
simlab.plotmesh(Th,d=2,color='LightGray',opacity=0.05)
simlab.slice(Th,u,origin=(0,0,1),normal=(-1,0,1))
mlab.view(132,53,7)
mlab.figure(2)
simlab.plotmesh(Th,d=2,color='LightGray',opacity=0.05)
for normal in [(1,0,0),(0,1,0),(0,0,1)]:
    simlab.slice(Th,u,origin=(0,0,1),normal=normal)
mlab.view(155,66,7)
mlab.figure(3)
simlab.plotmesh(Th,d=2,color='LightGray',opacity=0.05)
for normal in [(1,0,0),(0,1,0),(0,0,1)]:
    simlab.slice(Th,u,labels=[1],origin=(0,0,1),normal=normal)
mlab.view(155,66,7)
mlab.figure(4)
simlab.plotmesh(Th,d=2,color='LightGray',opacity=0.05)
for z in [0,0.5,1,1.5]:
    simlab.slice(Th,u,labels=[1],origin=(0,0,z),normal=(-1,0,1),opacity=0.6)
mlab.view(110,71,7)

```

Listing 15: 3D mesh : slice function

1.6 function SLICEISO

The `VTK_SLICEISO` function displays isolines of data on the intersection of a plane and a 3D mesh or parts of a 3D mesh defined by an `siMESH` object.

Syntaxe

```

simlab.sliceiso(Th,u,P)
simlab.sliceiso(Th,u,Key=Value,...)

```

Description

`simlab.sliceiso (Th,u)` displays `u` data as isolines on the intersection of a plane and all the 3-dimensional simplices elements. By default the plane is given by an *origin* point $(0, 0, 0)$ which lies on it and a *normal* vector $(0, 0, 1)$ which is orthogonal to the plane.

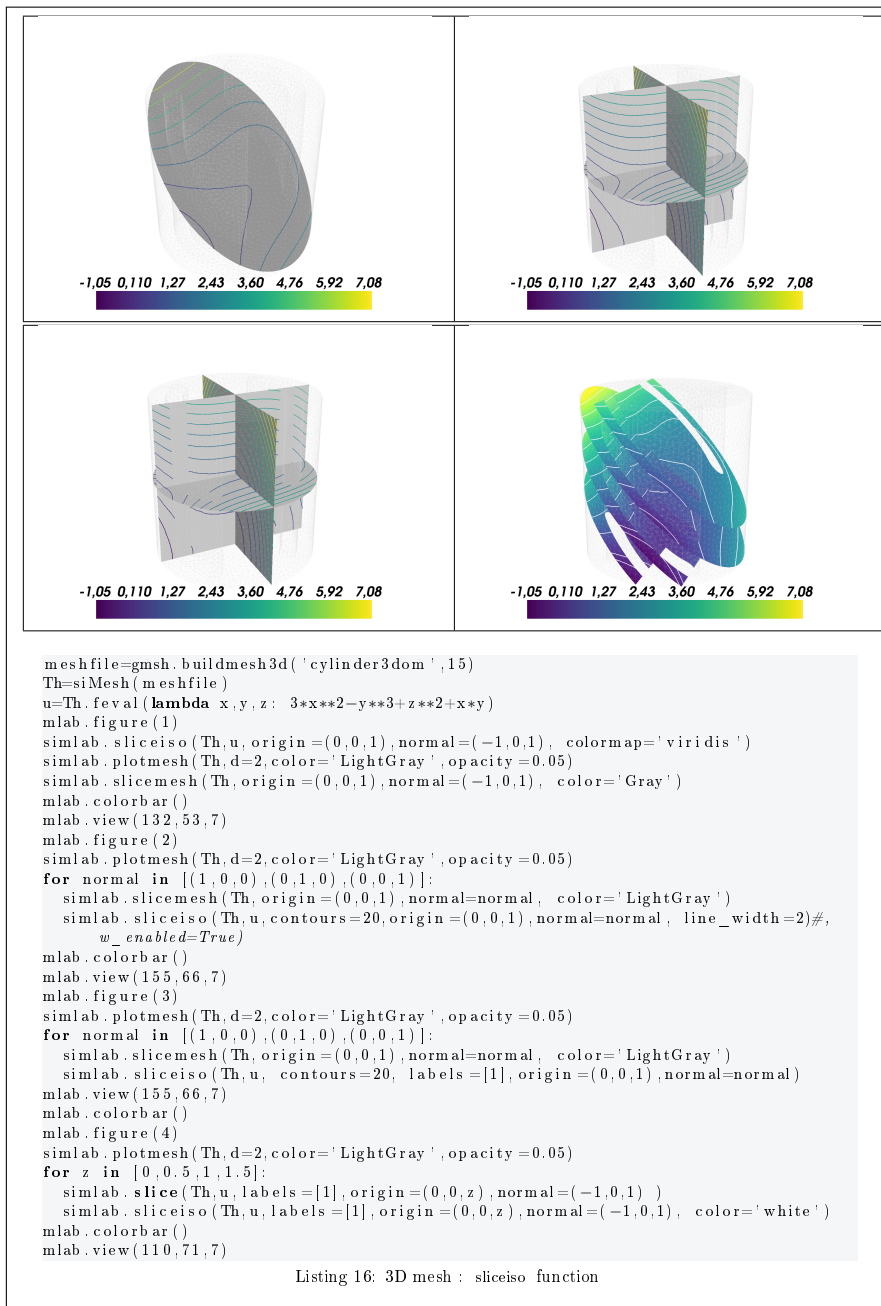
`simlab.sliceiso (Th,u,key=value, ...)` allows additional key/value pairs to be used when displaying `u`. The key could be

- `labels` : to select the labels of the elements to intersect,
- `contours` : to specify the number of isolines (default : 10) or a list/numpy array of isovalues (default : empty)
- `color` : to specify one color for all isolines (default : empty)

For key/value pairs, one could also used those of the `iso_surface` function from `mayavi.mlab.pipeline`.

1.6.1 3D example

The following example use the `.geo` file `ball8.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.



1.7 function `STREAMLINE`

The `STREAMLINE` function allows to draw streamlines for given vector data and colored by scalar data on a 3D mesh or parts of a 3D mesh defined by an `siMESH` object. This supports various types of seed objects (line, sphere, plane and point seeds). It also allows to draw ribbons or tubes and further supports different types of interactive modes of calculating the streamlines.

Syntaxe

```
simlab.streamline(Th,u,V)
simlab.streamline(Th,u,V,Key=Value, ...)
```

Description

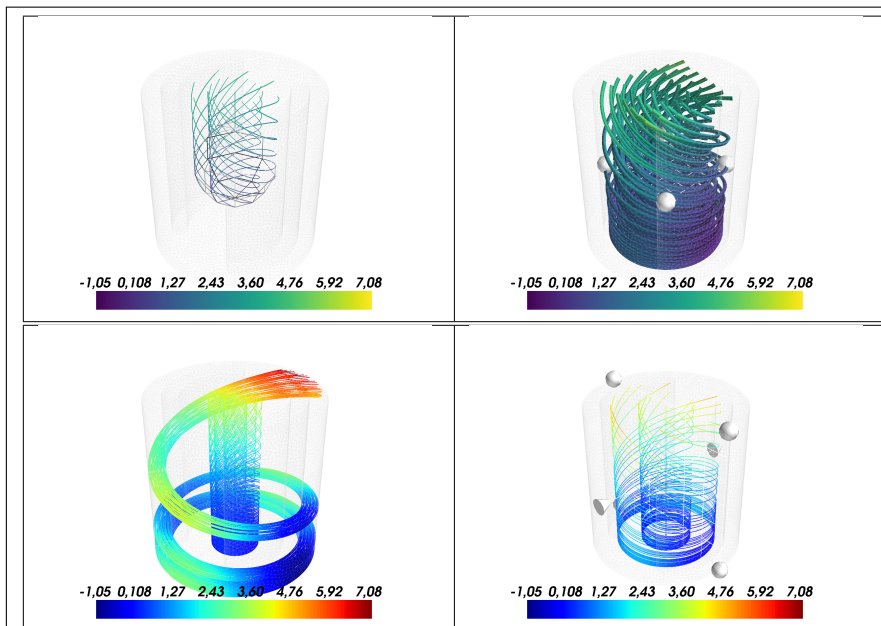
`simlab.streamline(Th,u,V)` displays streamlines computed from the vector data `V` and colored by the scalar data `u`

`simlab.streamline(Th,u,V,Key=Value, ...)` specifies function options using one or more `Key,Value` pair arguments. `Key` could be:

- `labels` : to select the labels of the `siMeshElt` on which to draw the streamlines.
- `seed_options` :
- `seed_widget_options` :
- `streamtracer_options` :

The other `Key/Value` pair options are those of `streamline` function from `mayavi.mlab.pipeline`.

The following example use the `.geo` file `cylinder3dom.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.



```

import numpy as np
meshfile=gmesh.buildmesh3d('cylinder3dom',20)
mlab.close(all=True)
Th=siMesh(meshfile)
U=Th.feval(lambda x,y,z: 3*x**2-y**3+z**2+x*y)
w=[lambda x,y,z: y*np.cos(-(x**2+y**2)/10), lambda x,y,z:
-x*np.cos(-(x**2+y**2)/10), lambda x,y,z: z/5]
W=np.array([Th.feval(w[0]),Th.feval(w[1]),Th.feval(w[2])])
mlab.figure(1)
simlab.streamline(Th,U,W)
simlab.plotmesh(Th,d=2,color='LightGray',opacity=0.05)
mlab.colorbar()
mlab.figure(2)
s_options={'visible':True}
sw_options={'normal':(0,0,1),'resolution':6}
st_options={'integration_direction':'both'}
s1=simlab.streamline(Th,U,W,seedtype='plane',linetype='tube',
                    seed_options=s_options,
                    seed_widget_options=sw_options,
                    streamtracer_options=st_options)
simlab.plotmesh(Th,d=2,color='LightGray',opacity=0.05)
mlab.colorbar()

mlab.figure(3)
sw_options={'center':(0.9,0,1),'radius':0.1,'phi_resolution':8,
            'theta_resolution':12,'enabled':False}
st_options={'integration_direction':'both'}
s12=simlab.streamline(Th,U,W,seed_widget_options=sw_options,
                     streamtracer_options=st_options,colormap='jet')
sw_options['center']=(0,0,1)
sw_options['radius']=0.3
s12=simlab.streamline(Th,U,W,seed_widget_options=sw_options,
                     streamtracer_options=st_options,colormap='jet')
mlab.scalarbar()
mlab.view(46.6,58,6.7)
simlab.plotmesh(Th,d=2,color='LightGray',opacity=0.05)

mlab.figure(4)
sw_options={'origin':(0,-1,0),'point1':(0,-1,2),'point2':(0,1,0),
            'enabled':True,'resolution':6}
st_options={'integration_direction':'both'}
s11=simlab.streamline(Th,U,W,seedtype='plane',
                    seed_widget_options=sw_options,
                    streamtracer_options=st_options,colormap='jet')
mlab.scalarbar()
mlab.view(46.6,58,6.7)
simlab.plotmesh(Th,d=2,color='LightGray',opacity=0.05)

```

Listing 17: 3D mesh : streamline function