



FC_SIMESH_MATPLOTLIB package, User's Guide *

François Cuvelier[†]

May 23, 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_MATPLOTLIB 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 matplotlib 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	3
1.1.3	3D surface example	4
1.2	function <code>PLOT</code>	5
1.2.1	2D example	6
1.2.2	3D example	7
1.2.3	3D surface example	8
1.3	function <code>PLOTISO</code>	9
1.3.1	2D example	10

*Compiled with Python 3.6.0, packages FC_HYPERMESH-0.0.4, FC_OOGMSH-0.0.4, FC_TOOLS-0.0.10, FC_SIMESH-0.0.6 and the plotting libraries MATPLOTLIB-2.0.0, FC_SIMESH_MATPLOTLIB-2.0.0

[†]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	function QUIVER	10
1.3.3	2D example	11
1.3.4	3D example	12
1.3.5	3D surface example	13

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

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

Description

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

`siplt.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)

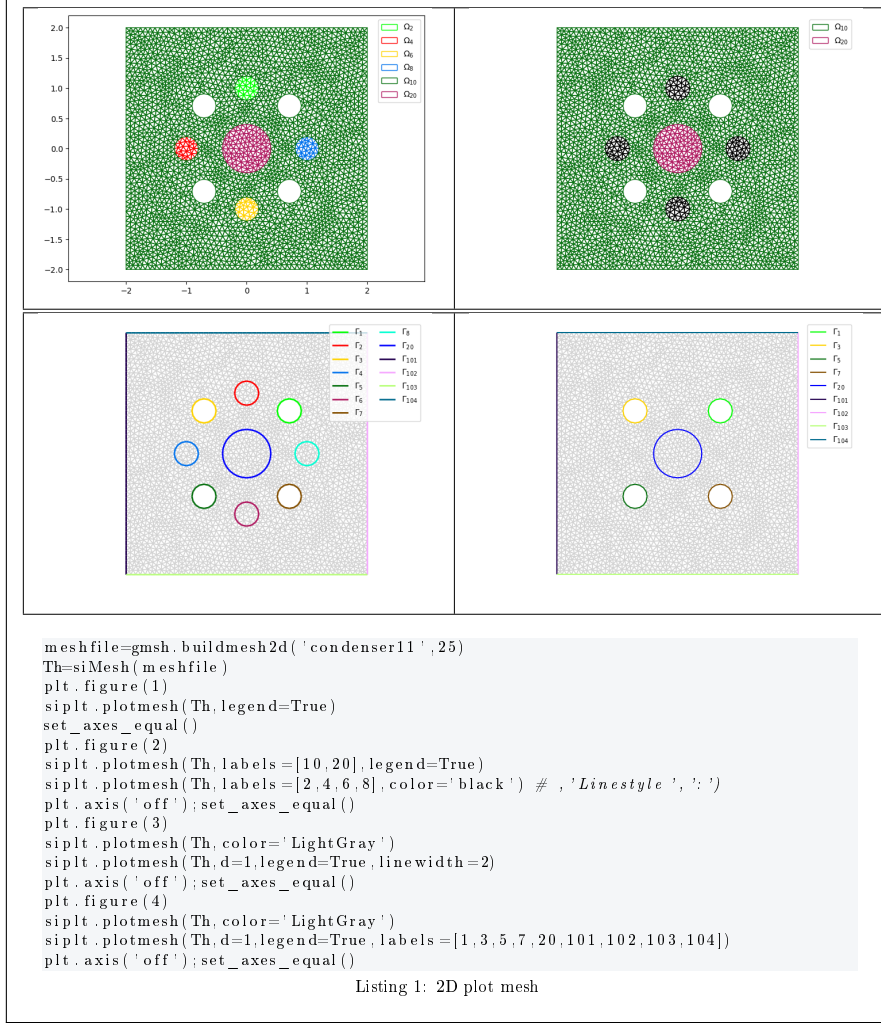
The options of second level depend on the type of elementaries mesh elements to represent.

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

- In dimension 3,
 - if $d == 3$, `Line3DCollection` function is used
 - if $d == 2$, `Poly3DCollection` function is used
 - if $d == 1$, `Line3DCollection` function is used
 - if $d == 0$, `ax.scatter` function is used
- In dimension 2,
 - if $d == 2$, `plt.triplot` function is used,
 - if $d == 1$, `ax.plot` function is used
 - if $d == 0$, `ax.plot` function is used
- dimension 1, not yet implemented

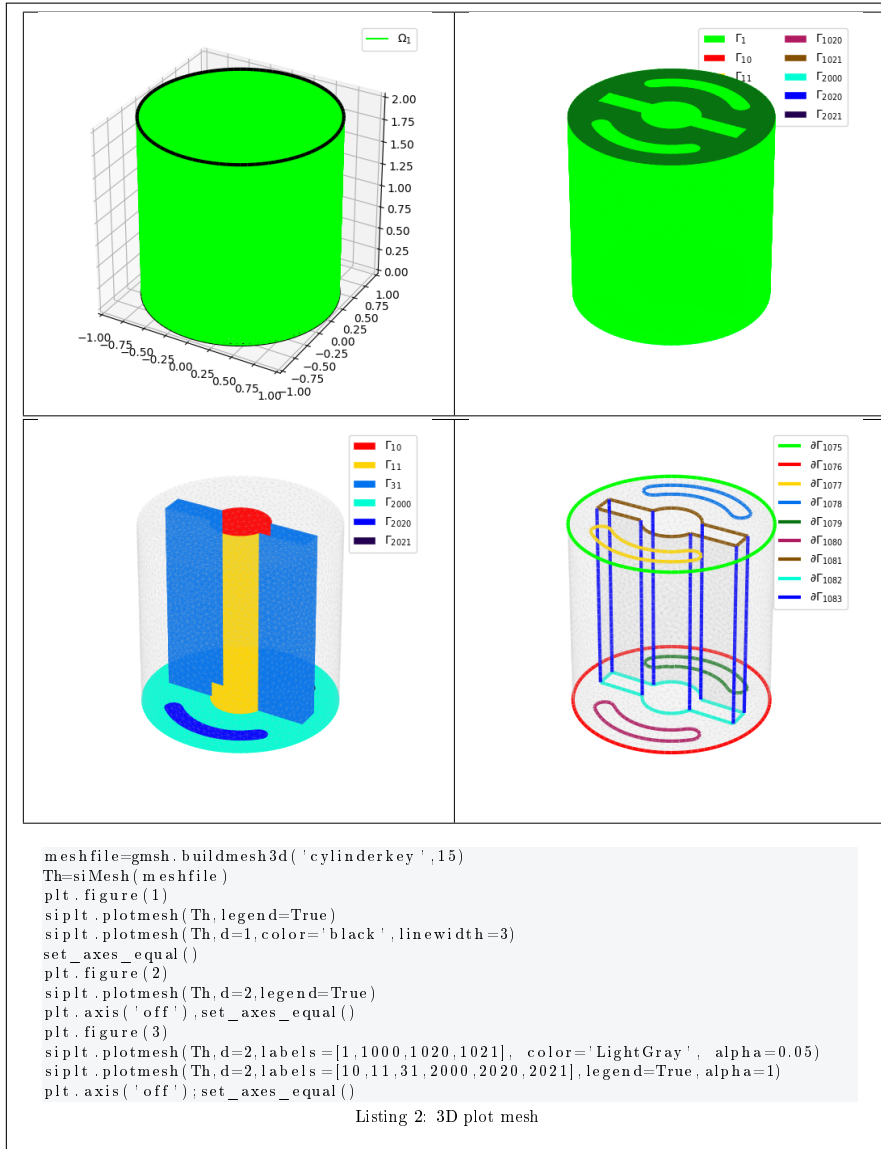
1.1.1 2D example

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



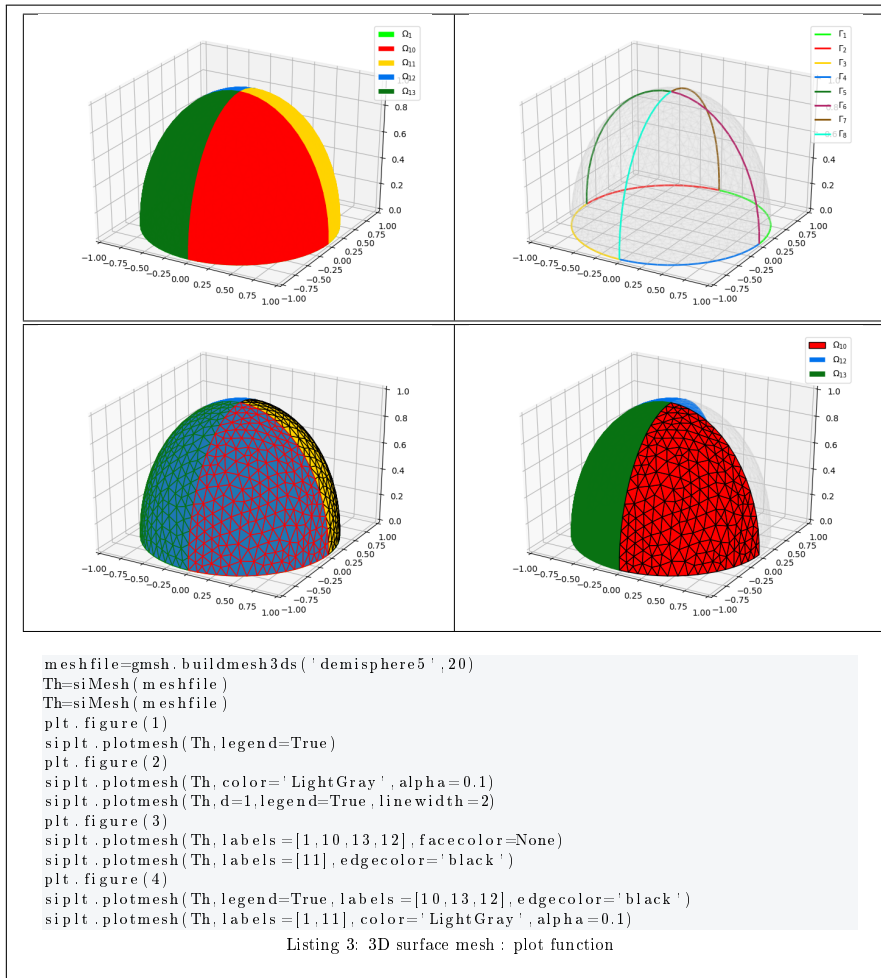
1.1.2 3D example

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 datas on the mesh or parts of the mesh defined by an **siMESH** object.

Syntaxe

```

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

```

Description

`siplt.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`.

`siplt.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)

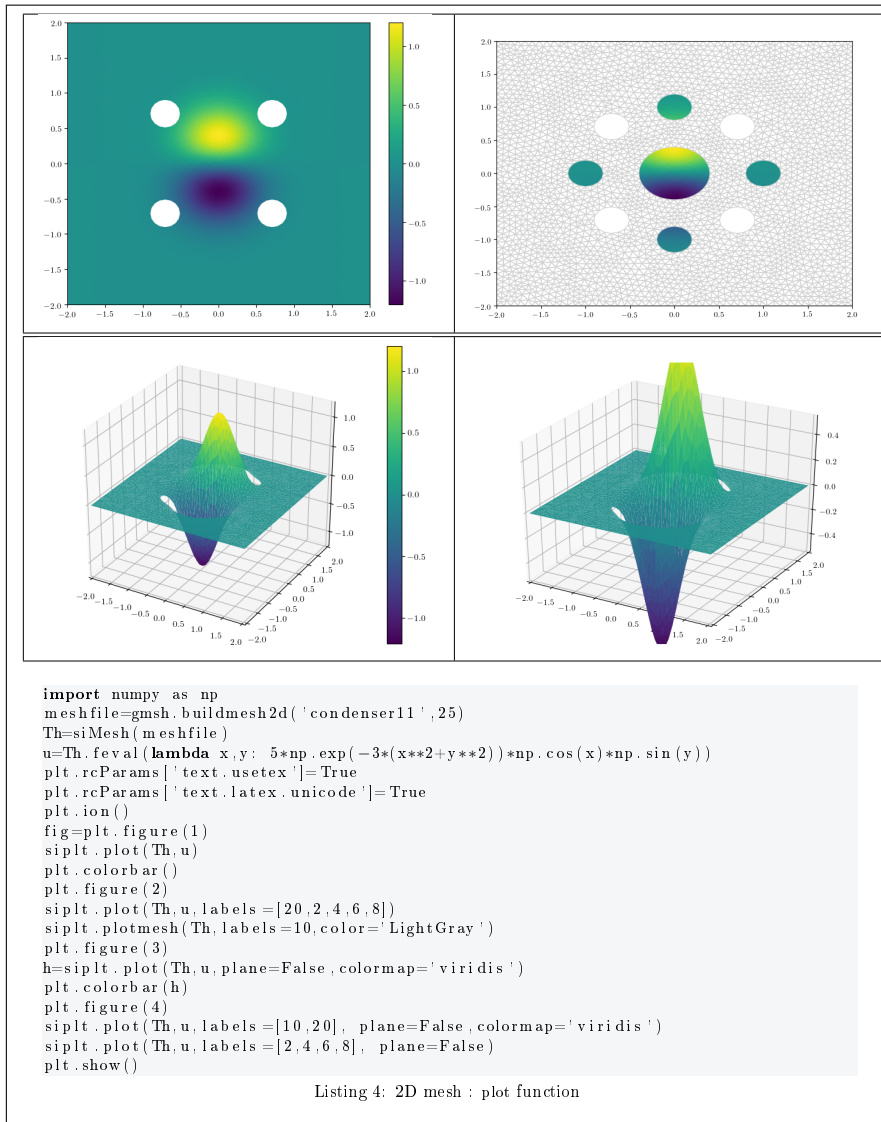
The second level options depend on the type of elementaries mesh elements on which we want to represent datas.

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

- In dimension 3,
 - if $d == 3$, `scatter3D` function is used
 - if $d == 2$, `Poly3DCollection` function is used.
 - if $d == 1$, `Line3DCollection` function is used.
- In dimension 2,
 - if $d == 2$, `tripcolor` function is used.
 - if $d == 1$, `LineCollection` function is used.
- Dimension 1 : not implemented.

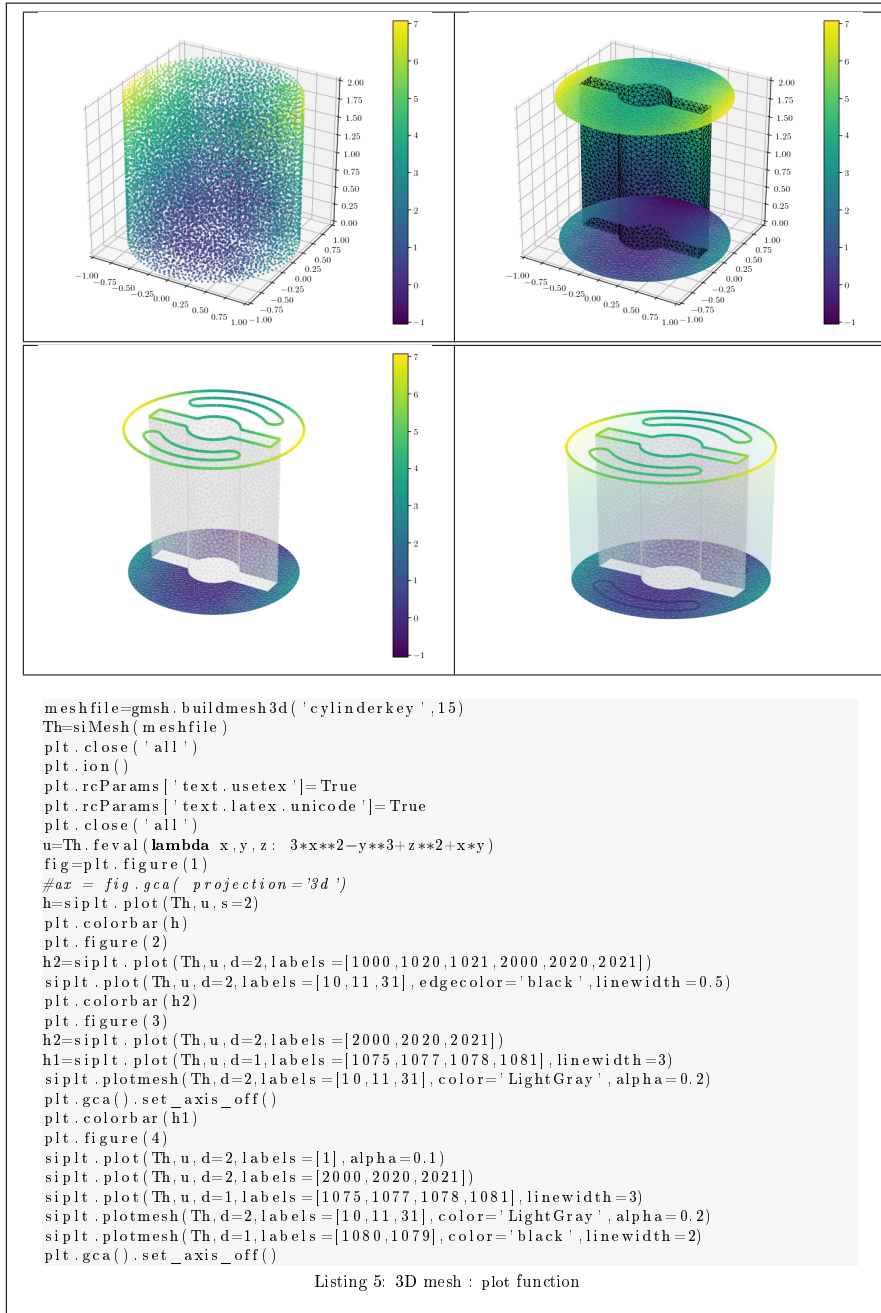
1.2.1 2D example

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



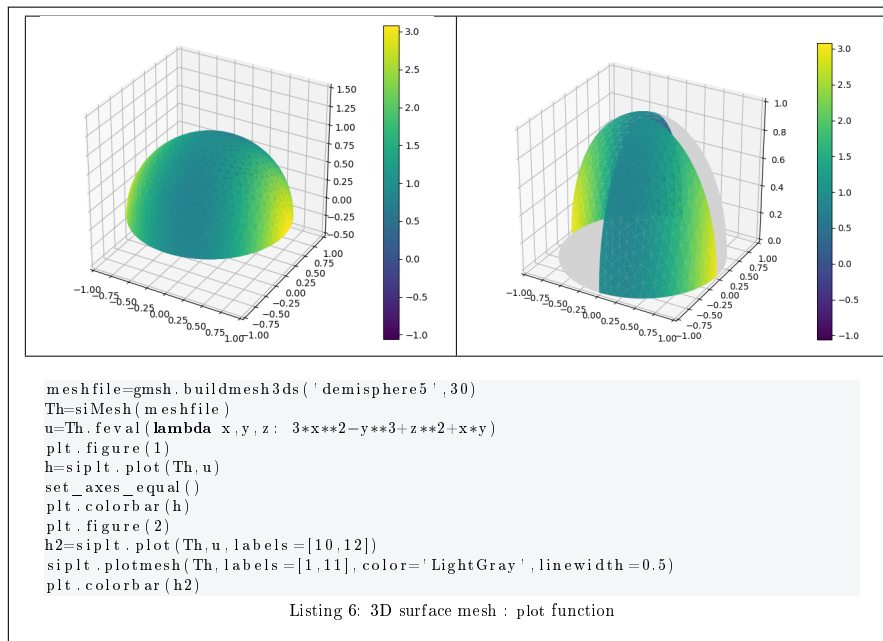
1.2.2 3D example

The following example 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.



1.2.3 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.

Syntaxe

```

sipplt.plotiso(Th,u)
sipplt.plotiso(Th,u,Key=Value, ...)

```

Description

`sipplt.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`.

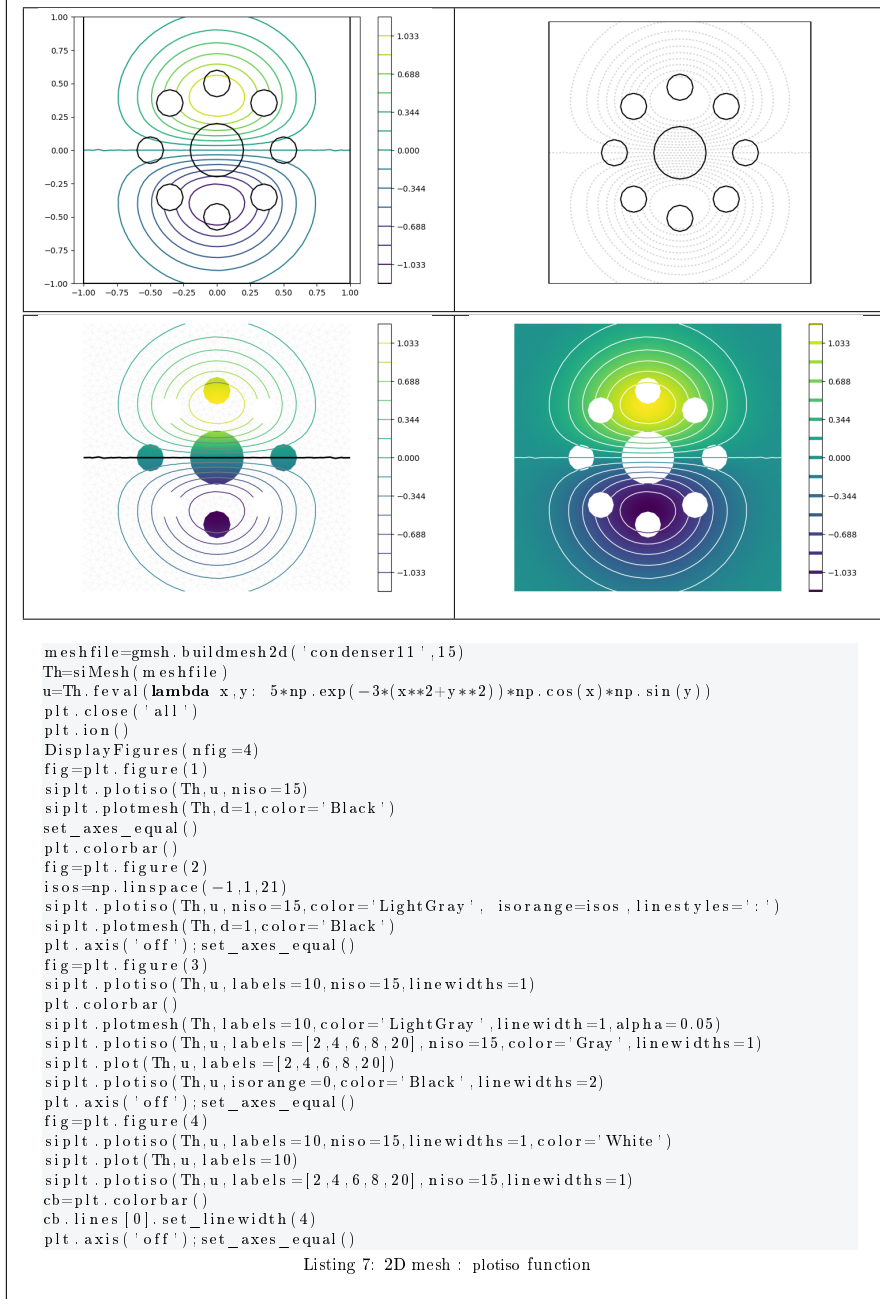
`sipplt.plotiso(Th,u,key=value, ...)` specifies function options using one or more key,value pair arguments. Options of first level are

- **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,
- **colormap** : to specify the name of the colormap to use (default : 'viridis')
- **color** : to specify one color for all isolines (default : None)

The second level options are the options of the `plt.tricontour` 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 function **QUIVER**

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

Syntaxe

```
siplt.quiver(Th,V)
siplt.quiver(Th,V,Key=Value, ...)
```

Description

`siplt.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- $Th.nq$.

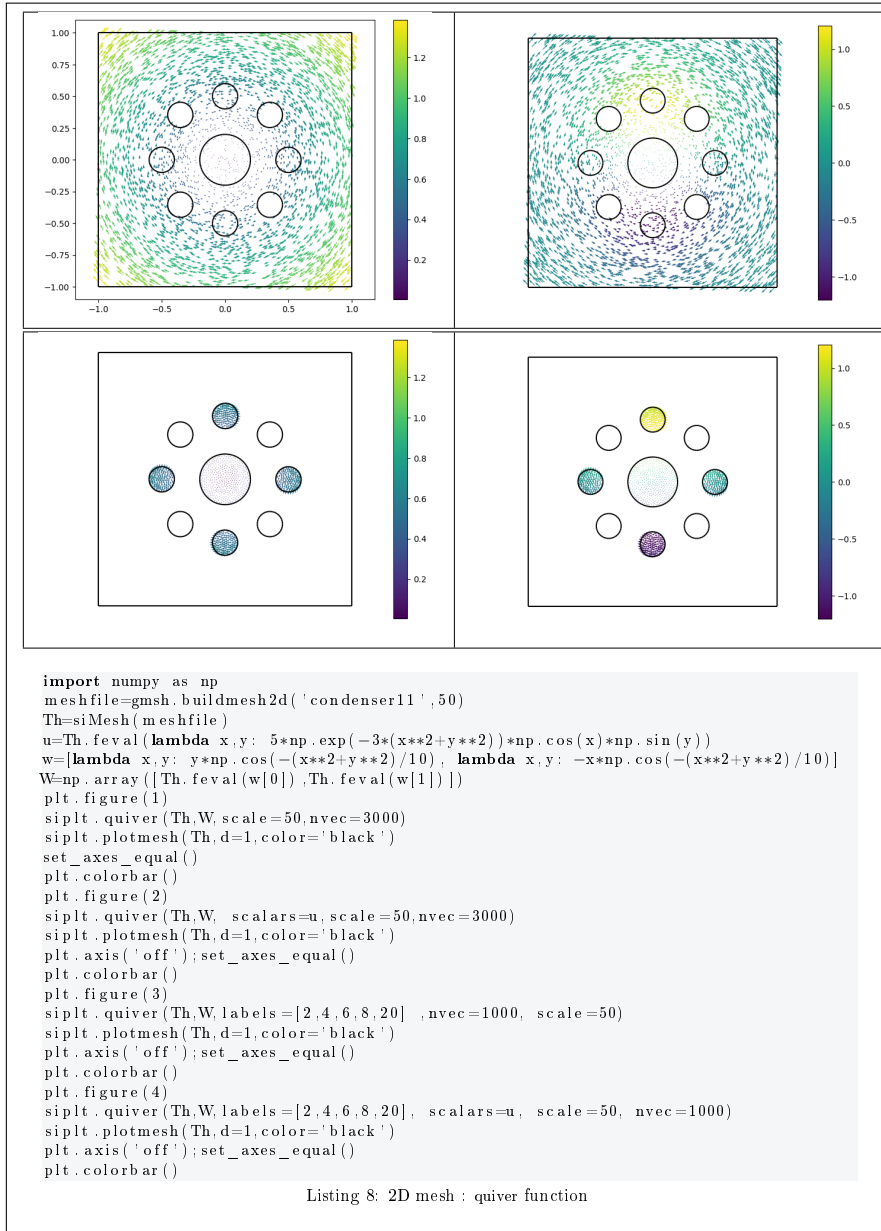
`siplt.quiver(Th,V,Key=Value, ...)` specifies function options using one or more Key,Value pair arguments. Options of first level are

- **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. By default colored with the Euclidian norm of the vectors.

For key/value pairs, one could also used those of the `plt.quiver` function in dimension 2 and `mpl_toolkits.mplot3d.quiver` function in dimension 3

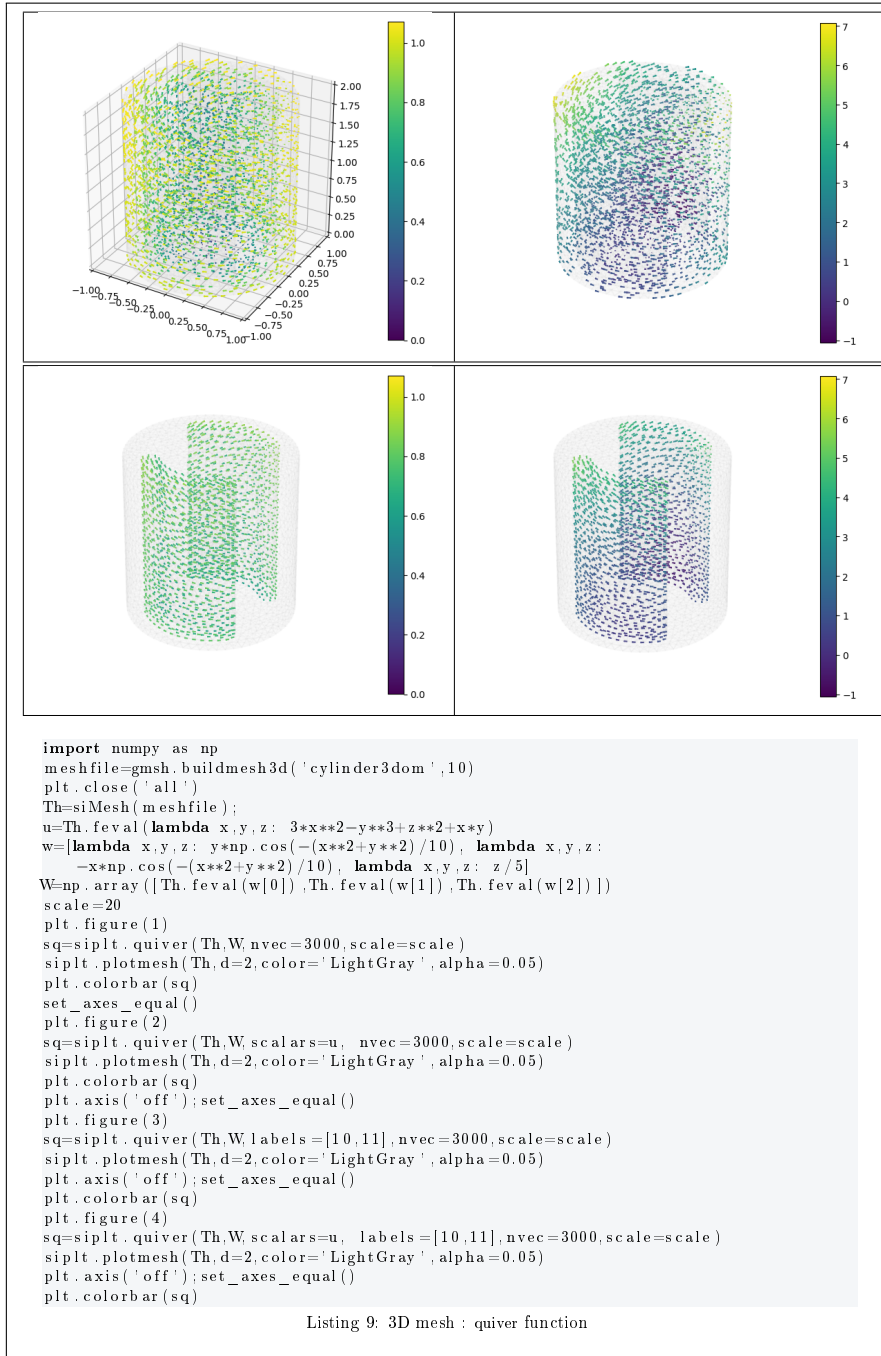
1.3.3 2D example

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



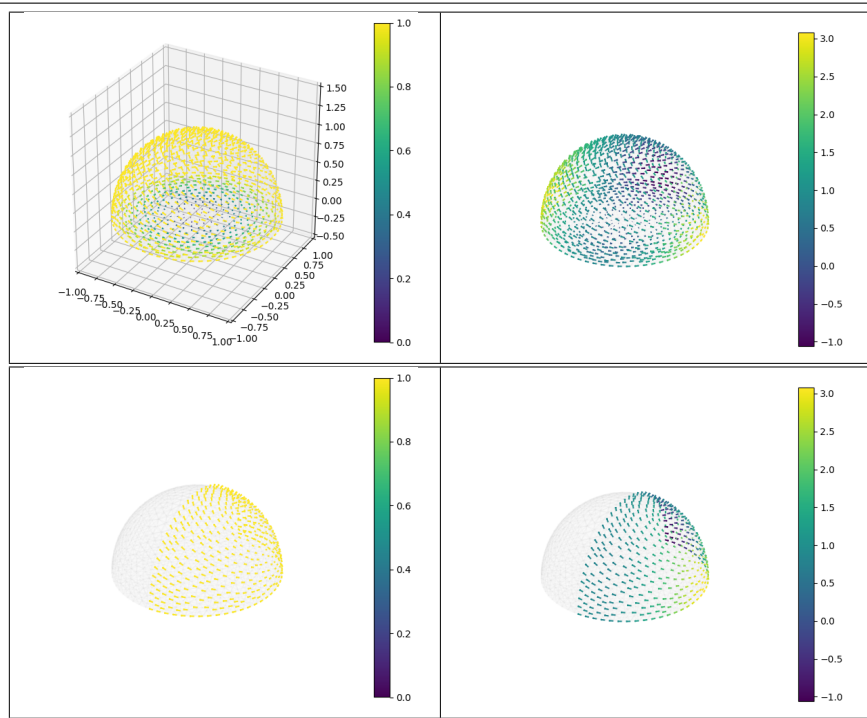
1.3.4 3D example

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.



1.3.5 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.



```
import numpy as np
meshfile=gmsl.buildmesh3ds('demisphere5',30)
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]
W=np.array([Th.feval(w[0]),Th.feval(w[1]),Th.feval(w[2])])
scale=20
plt.figure(1)
sq=siplt.quiver(Th,W,nvec=3000,scale=scale)
siplt.plotmesh(Th,d=2,color='LightGray',alpha=0.05)
plt.colorbar(sq)
set_axes_equal()
plt.figure(2)
sq=siplt.quiver(Th,W,scalars=u,nvec=3000,scale=scale)
siplt.plotmesh(Th,d=2,color='LightGray',alpha=0.05)
plt.colorbar(sq)
plt.axis('off');set_axes_equal()
plt.figure(3)
sq=siplt.quiver(Th,W,labels=[10,11],nvec=3000,scale=scale)
siplt.plotmesh(Th,d=2,color='LightGray',alpha=0.05)
plt.axis('off');set_axes_equal()
plt.colorbar(sq)
plt.figure(4)
sq=siplt.quiver(Th,W,scalars=u,labels=[10,11],nvec=3000,scale=scale)
siplt.plotmesh(Th,d=2,color='LightGray',alpha=0.05)
plt.axis('off');set_axes_equal()
plt.colorbar(sq)
```

Listing 10: 3D surface mesh : quiver function