matplotlib4mesh Python package, User's Guide*

version 0.1.2

F. Cuvelier†

May 14, 2019

Abstract

The matplotlib4mesh Python package allows to display simplices meshes or datas on simplices meshes by using Matplotlib (>=2.0.0) Python package. The simplicial meshes must be given by two arrays : the vertices array and the connectivity array.

0 Contents

*LaTeX manual, revision 0.1.2, compiled with Python 3.7.3, and toolboxes `fc-matplotlib4mesh`[0.1.2], `fc-tools`[0.0.22], `fc-meshtools`[0.1.1],

†LAGA, UMR 7539, CNRS, Université Paris 13 - Sorbonne Paris Cité, Université Paris 8, 99 Avenue J-B Clément, F-93430 Villetaneuse, France, cuvelier@math.univ-paris13.fr

1 Introduction

The **experimental** toolbox package use Matplotlib (>=2.0.0) Python package for displaying simplicial meshes or datas on simplicial meshes. Simplicial meshes could be:

- a triangular mesh in dimension 2, made with 2-simplices (ie. triangles),
- a tetrahedral mesh in dimension 3, made with 3-simplices (ie. tetrahedron),
- a triangular mesh in dimension 3 (surface mesh), made with 2-simplices,
- a line mesh in dimension 2 or 3 made with 1-simplices (ie. lines).

A simplicial mesh is given by its vertices array q and its connectivity array me. For demonstration purpose, we used some simplicial meshes provided by the `fc_meshtools` Python package: they can be load by using the function `getMesh2D`, `getMesh3D` or `getMesh3Ds` of the `fc_meshtools.simplicial` module.

remark 1.1

The **fc_matplotlib4mesh** package can be used on small meshes. For meshes with a large number of elements, for reasons of performance, the use of the **fc_mayavi4mesh** package [1] is highly recommended.

1.1 Installation

For Python 2, the installation of this package can be done with the `pip` command.

- For an installation which isolated to the current user, one can do:

```
$ pip install --user fc_matplotlib4mesh
```

- For an installation for all users, one can do:

```
$ sudo pip install fc_matplotlib4mesh
```

For Python 3 installation, sometimes `pip3` must be used instead of `pip`.

One can note that the **fc_tools** and **fc_meshtools** Python packages are automaticaly installed.

1.2 Remarks

In all this report, for graphical representation, we use `plt4sim` namespace for graphical functions using **matplotlib**. There is the python code previously used before any of the listings given thereafter:

```
import numpy as np
import matplotlib.pyplot as plt
import fc_matplotlib4mesh as plt4sim
from fc_meshtools.simplicial import getMesh2D,getMesh3D,getMesh3Ds
```

The functions `getMesh2D`, `getMesh3D` and `getMesh3Ds` return a mesh vertices array q, a mesh elements connectivity array and an indices array `toGlobal` array associated with the input argument d (simplex dimension). The vertices array q is a dim-by-n_q or n_q-by-dim numpy array where dim is the space dimension (2 or 3) and n_q the number of vertices. The connectivity array me is a $(d+1)$-by-n_{me} or n_{me}-by-$(d+1)$ numpy array where n_{me} is the number of mesh elements and $0 \leqslant d \leqslant dim$ is the simplicial dimension:

- $d = 1$: lines,
- $d = 2$: triangle,
- $d = 3$: tetrahedron.

For example, `q,me,toGlobal=getMesh3D(2)` return a 2-simplicial mesh in space dimension $dim = 3$.

2 function PLOTMESH

The PLOTMESH function displays a mesh given by a vertices and connectivity arrays.

Syntaxe

```
plt4sim.plotmesh(q,me)
plt4sim.plotmesh(q,me,Key=Value, ...)
```

Description

`plt4sim.plotmesh(q,me,)` displays all the `d`-dimensional simplices elements.

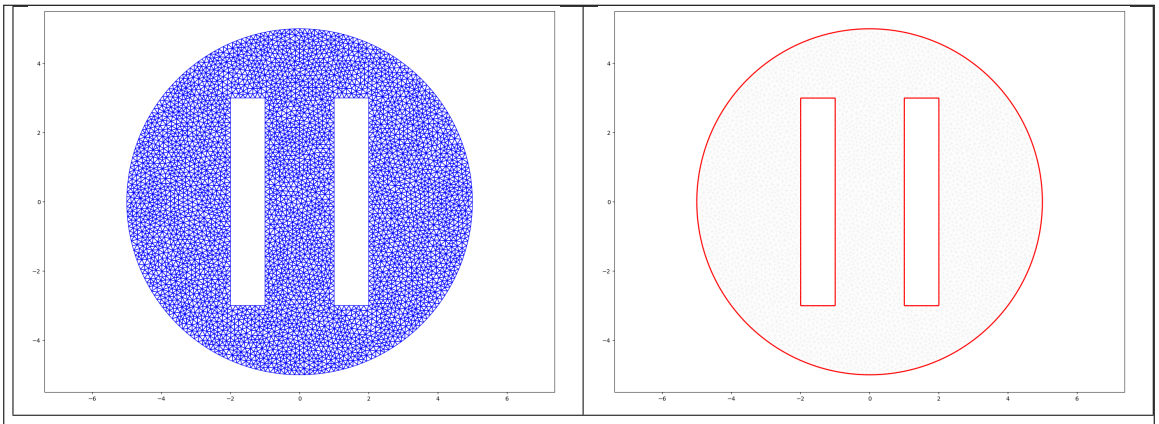
`plt4sim.plotmesh(q,me,Key=Value, ...)` specifies function options using one or more `Key`,`Value` pair arguments. Options of first level are

- `color` : specify the mesh color as a RGB tupple or a string color (see check_color ...)
- `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 $d == 3$, `Line3DCollection` function is used with `add_collection` and `tvtk.Tetra().cell_type`
  - if $d == 2$, `Poly3DCollection` function is used with `add_collection`,
  - if $d == 1$, `Line3DCollection` function is used with `add_collection`,
  - if $d == 0$, `Axes3D.scatter` function is used.
- In dimension 2,
  - if $d == 2$, `PolyCollection` function is used with `add_collection`,
  - if $d == 1$, if $d == 1$, `LineCollection` function is used with `add_collection`,
  - if $d == 0$, `matplotlib.pyplot.plot` function is used
- dimension 1, not yet implemented

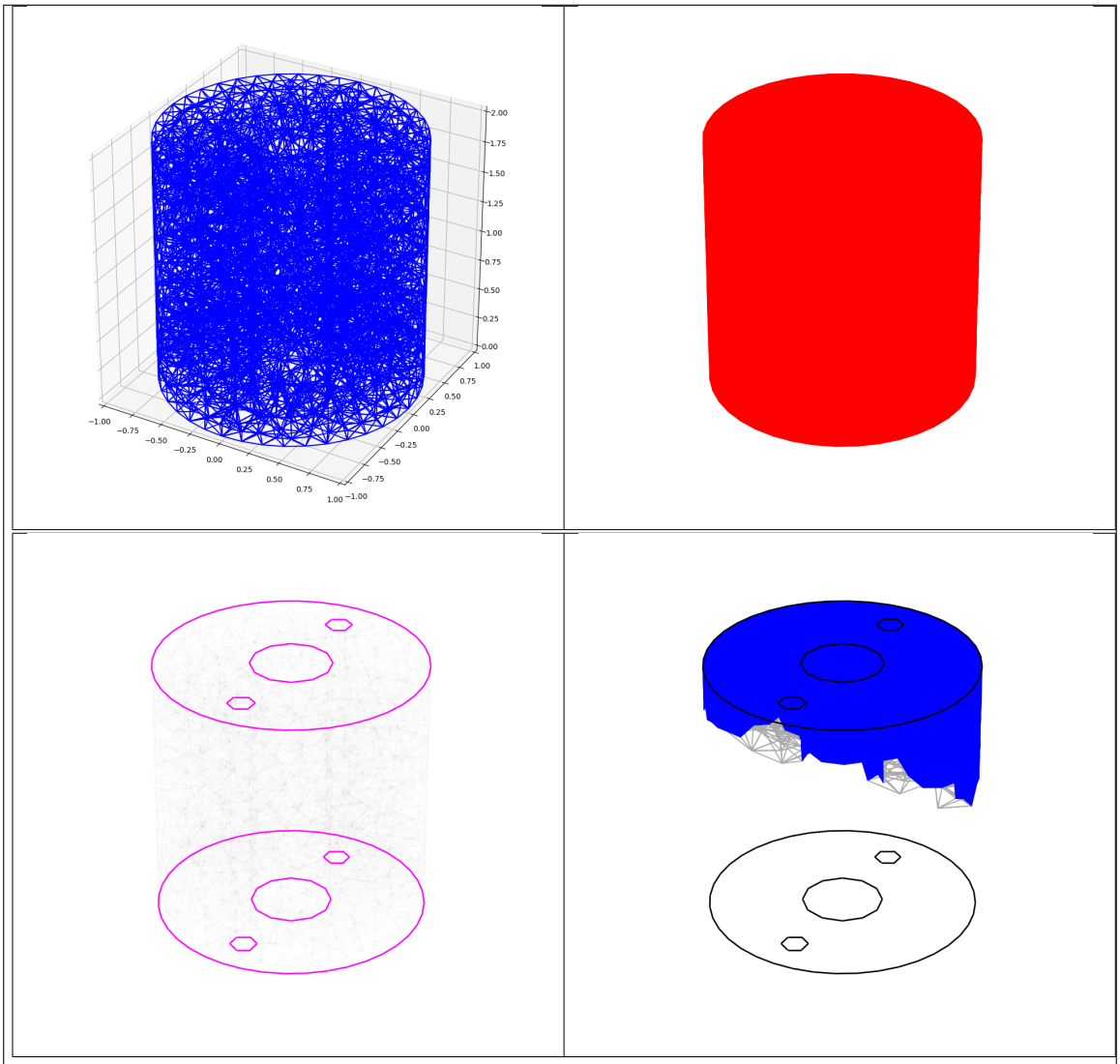
2.1 2D example



```
plt.ion()
q2,me2,toG2=mshsim.getMesh2D(2)
q1,me1,toG1=mshsim.getMesh2D(1)
plt.ion()
plt.close('all')
plt.figure(1)
plt4sim.plotmesh(q2,me2)
mpltools.set_axes_equal()
plt.figure(2)
plt4sim.plotmesh(q1,me1,color='Red',linewidth=2)
plt4sim.plotmesh(q2,me2,color='LightGray',alpha=0.1)
mpltools.set_axes_equal()
```

Listing 1: 2D plot mesh

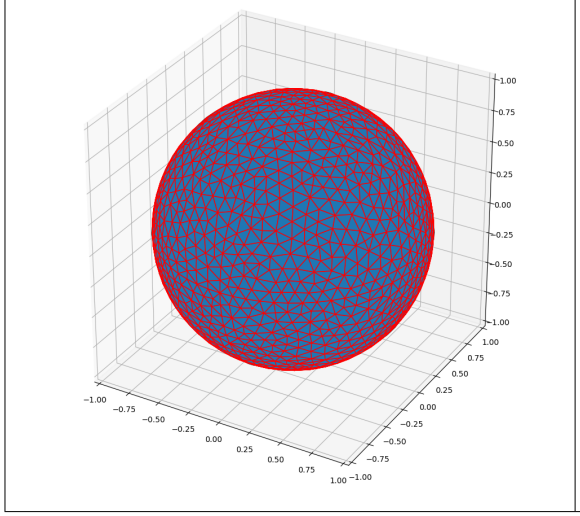
2.2 3D example

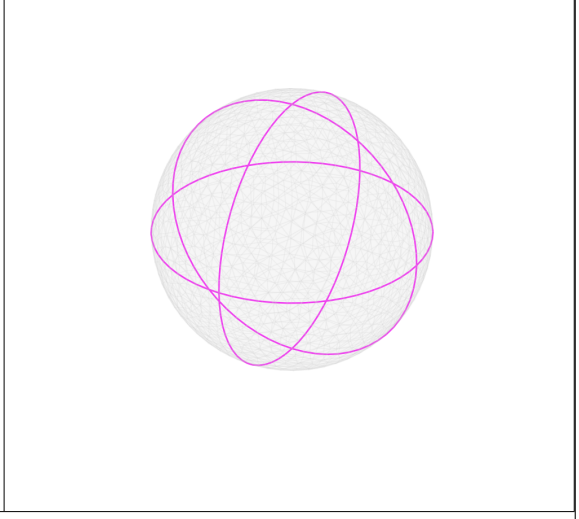


```
q3,me3,toG3=mshsim.getMesh3D(3,small=True)
q2,me2,toG2=mshsim.getMesh3D(2,small=True)
q1,me1,toG1=mshsim.getMesh3D(1,small=True)
plt.ion()
plt.figure(1)
plt4sim.plotmesh(q3,me3)
mpltools.set_axes_equal()
plt.figure(2)
plt4sim.plotmesh(q2,me2,color='Red')
mpltools.set_axes_equal()
plt.axis('off')
plt.figure(3)
plt4sim.plotmesh(q3,me3,color='LightGray',alpha=0.02)
plt4sim.plotmesh(q1,me1,color='Magenta',linewidths=2)
from fc_tools.graphics import Plane
P=[Plane(origin=[0,0,1],normal=[0,0,-1]), Plane(origin=[0,0,1],normal=[0,-1,-1])]
mpltools.set_axes_equal()
plt.axis('off')
plt.figure(4)
plt4sim.plotmesh(q3,me3,cut_planes=P,color='DarkGrey')
plt4sim.plotmesh(q2,me2,cut_planes=P)
plt4sim.plotmesh(q1,me1,color='Black',linewidths=2)
mpltools.set_axes_equal()
plt.axis('off')
```

Listing 2: 3D plot mesh

2.3 3D surface example





```
q2,me2,toG2=mshsim.getMesh3Ds(2)
q1,me1,toG1=mshsim.getMesh3Ds(1)
plt.ion()
plt.figure(1)
plt4sim.plotmesh(q2,me2,edgecolor='Red',facecolor=None)
plt4sim.plotmesh(q1,me1,color='Black',linewidths=2)
mpltools.set_axes_equal()
plt.figure(2)
plt4sim.plotmesh(q2,me2,color='LightGray',alpha=0.1)
plt4sim.plotmesh(q1,me1,color='Magenta',linewidths=2)
plt.axis('off')
mpltools.set_axes_equal()
```

Listing 3: 3D surface mesh : plot function

3 function PLOT

The PLOT function displays scalar datas on a mesh given by vertices and connectivity arrays.

Syntaxe

```
plt4sim.plot(q,me,u)
plt4sim.plot(q,me,u,Key=Value, ...)
```

Description

`plt4sim.plot(q,me,u)` displays data `u` on the given mesh. The data `u` is an 1D-array of size n_q

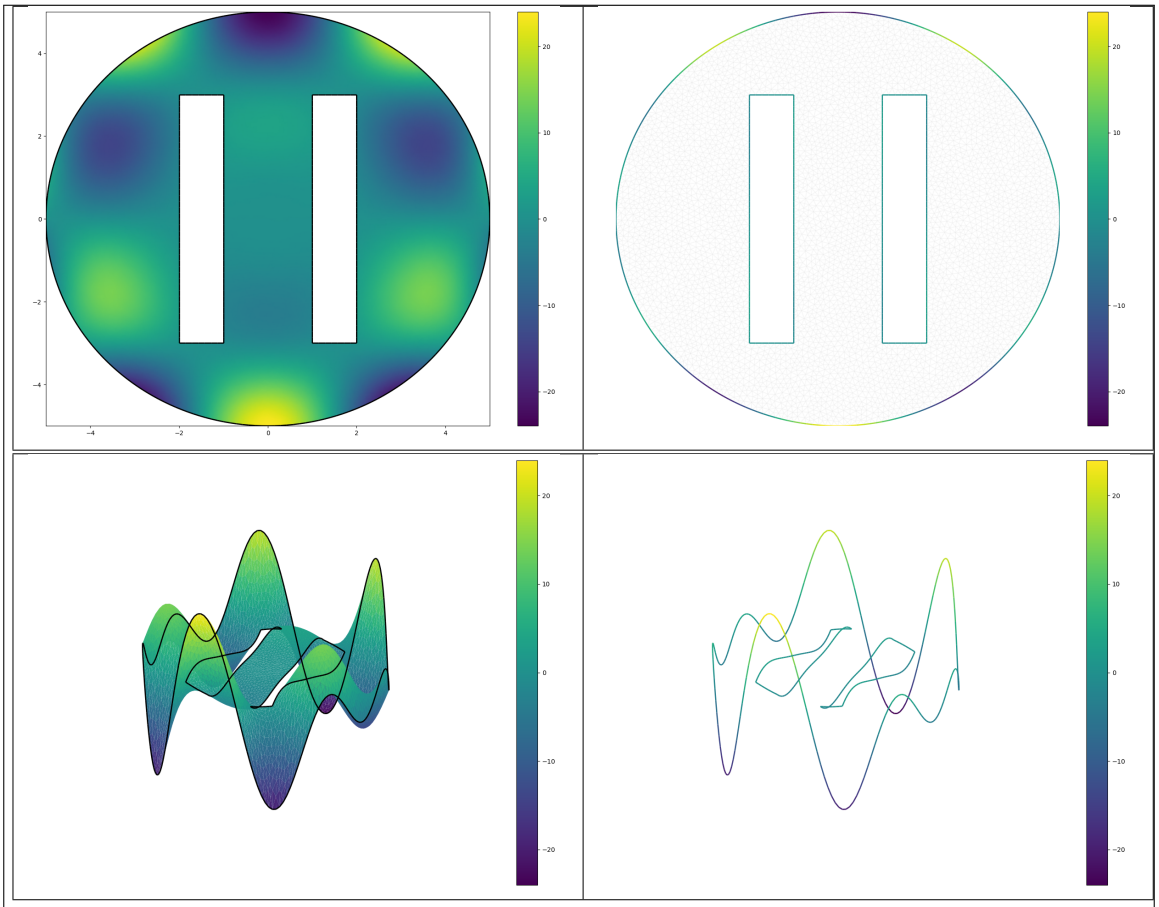
`plt4sim.plot(q,me,u,Key=Value, ...)` specifies function options using one or more `Key`,`Value` pair arguments. Options of first level are

- `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 $d == 3$, `Axe3D.scatter3D` function is used,
  - if $d == 2$, `Poly3DCollection` function is used with `add_collection`,
  - if $d == 1$, `Line3DCollection` function is used with `add_collection`.
- In dimension 2,
  - if $d == 2$, `plt.tripcolor` function is used,
  - if $d == 1$, `LineCollection` function is used with `add_collection`.
- Dimension 1 : not implemented.

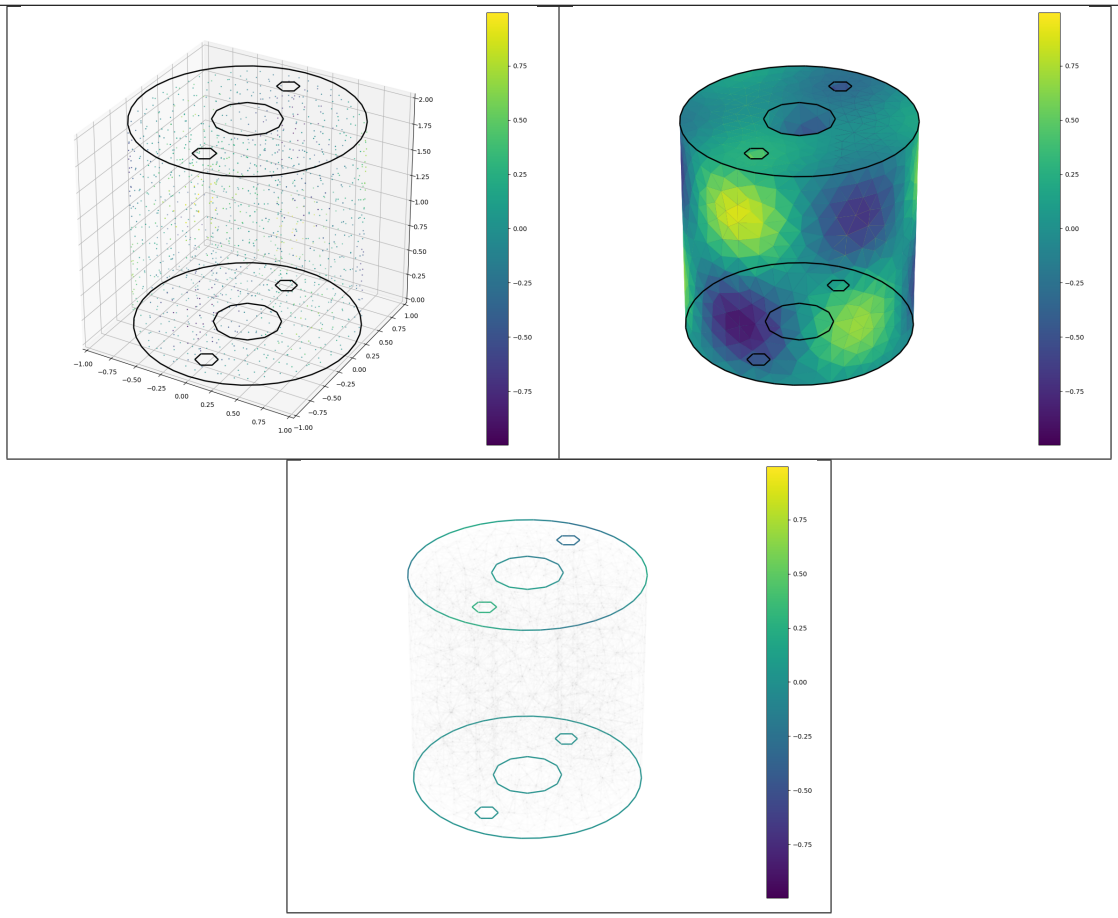
3.1 2D example



```
q2,me2,toG2=mshsim.getMesh2D(2)
q1,me1,toG1=mshsim.getMesh2D(1)
f=lambda x,y: (x**2+y**2)*np.cos(x)*np.sin(y)
u2=f(q2[0],q2[1])
u1=f(q1[0],q1[1])
plt.ion()
plt.figure(1)
ps1=plt4sim.plot(q2,me2,u2)
plt4sim.plotmesh(q1,me1,color='Black',linewidths=2)
plt.colorbar()
plt.figure(2)
ps2=plt4sim.plot(q1,me1,u1,linewidths=2)
plt.colorbar(ps2)
plt4sim.plotmesh(q2,me2,color='LightGray',alpha=0.1)
plt.axis('off')
plt.figure(3)
ps3=plt4sim.plot(q2,me2,u2,plane=False)
plt4sim.plotmesh(q1,me1,z=u1,color='Black',linewidths=2)
plt.colorbar(ps3)
plt.axis('off')
plt.figure(4)
ps4=plt4sim.plot(q1,me1,u1,linewidths=2,plane=False)
plt.colorbar(ps4)
plt.axis('off')
```

Listing 4: 2D mesh : `plot` function

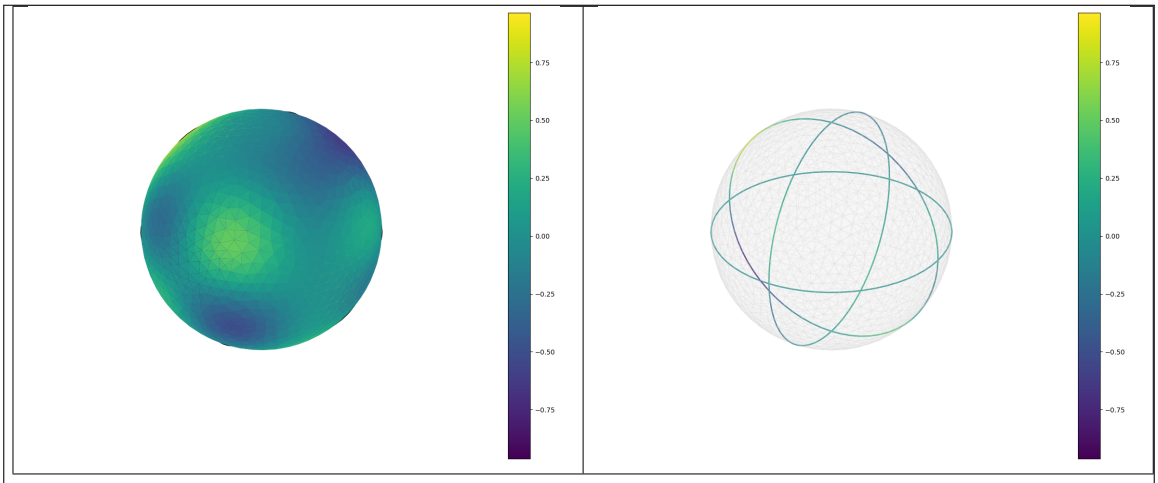
3.2 3D example



```
q3,me3,toG3=mshsim.getMesh3D(3,small=True)
q2,me2,toG2=mshsim.getMesh3D(2,small=True)
q1,me1,toG1=mshsim.getMesh3D(1,small=True)
f=lambda x,y,z: np.cos(3*x)*np.sin(2*y)*np.sin(3*z)
u3=f(q3[0],q3[1],q3[2])
u2=f(q2[0],q2[1],q2[2])
u1=f(q1[0],q1[1],q1[2])
plt.close('all')
plt.ion()
plt.figure(1)
pp=plt4sim.plot(q3,me3,u3)
plt4sim.plotmesh(q1,me1,color='Black',linewidths=2)
plt.colorbar(pp)
mpltools.set_axes_equal()
plt.figure(2)
pp=plt4sim.plot(q2,me2,u2)
plt4sim.plotmesh(q1,me1,color='Black',linewidths=2)
plt.axis('off')
mpltools.set_axes_equal()
plt.colorbar(pp)
plt.figure(3)
pp=plt4sim.plot(q1,me1,u1,linewidths=2,vmin=min(u3),vmax=max(u3))
plt4sim.plotmesh(q3,me3,color='LightGray',alpha=0.02)
plt.colorbar(pp)
plt.axis('off')
mpltools.set_axes_equal()
```

Listing 5: 3D mesh : `plot` function

3.3 3D surface example



```
q2,me2,toG2=mshsim.getMesh3Ds(2)
q1,me1,toG1=mshsim.getMesh3Ds(1)
f=lambda x,y,z: np.cos(3*x-1)*np.sin(2*y-2)*np.sin(3*z)
u2=f(q2[0],q2[1],q2[2])
u1=f(q1[0],q1[1],q1[2])
plt.close('all')
plt.ion()
plt.figure(1)
pp=plt4sim.plot(q2,me2,u2)
plt4sim.plotmesh(q1,me1,color='Black',linewidths=2)
plt.colorbar(pp)
plt.axis('off')
mpltools.set_axes_equal()
plt.figure(2)
pp=plt4sim.plot(q1,me1,u1,linewidths=2,vmin=min(u2),vmax=max(u2))
plt4sim.plotmesh(q2,me2,color='LightGray',alpha=0.1)
plt.colorbar(pp)
plt.axis('off')
mpltools.set_axes_equal()
```

Listing 6: 3D surface mesh : `plot` function

4 function PLOTISO

The PLOTISO function displays isolines from datas on a mesh given by a vertices and connectivity arrays. This function only works with 2-simplices in space dimension 2 or 3.

Syntaxe

```
plt4sim.plotiso(q,me,u)
plt4sim.plotiso(q,me,u,Key=Value, ...)
```

Description

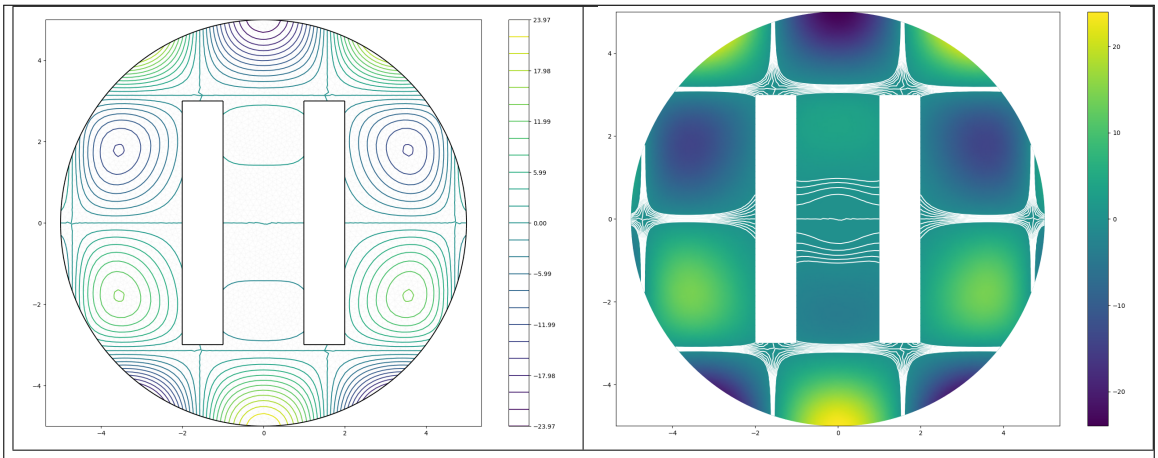
`plt4sim.plotiso(q,me,u)` displays data `u` on all the 2-dimensional simplices elements in dimension 2 as colored isovalues. The data `u` is an 1D-array of size `nq`.

`plt4sim.plotiso(q,me,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 : 15) or
- `isorange` : to specify a list/numpy array of isovalues (default : empty) (default : `True`)
- `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.

4.1 2D example



```
q2,me2,toG2=mshsim.getMesh2D(2)
q1,me1,toG1=mshsim.getMesh2D(1)
f=lambda x,y: (x**2+y**2)*np.cos(x)*np.sin(y)
u2=f(q2[0],q2[1])
plt.close('all')
plt.ion()
cmap=plt.cm.get_cmap(name='viridis')
plt4sim.plotiso(q2,me2,u2,niso=25,cmap=cmap)
plt.colorbar()
plt4sim.plotmesh(q1,me1,color='black')
plt4sim.plotmesh(q2,me2,color='LightGray',alpha=0.04)
mpltools.set_axes_equal()
plt.figure(2)
plt4sim.plot(q2,me2,u2,cmap=cmap)
plt.colorbar()
mpltools.set_axes_equal()
plt4sim.plotiso(q2,me2,u2,isorange=np.arange(-1,1,0.2),color='White')
```

Listing 7: 2D mesh : `plotiso` function

5 function QUIVER

The QUIVER function displays vector field datas on a mesh given by a vertices and connectivity arrays.

Syntaxe

```
plt4sim.quiver(q,me,V)
plt4sim.quiver(q,me,V,Key=Value, ...)
```

Description

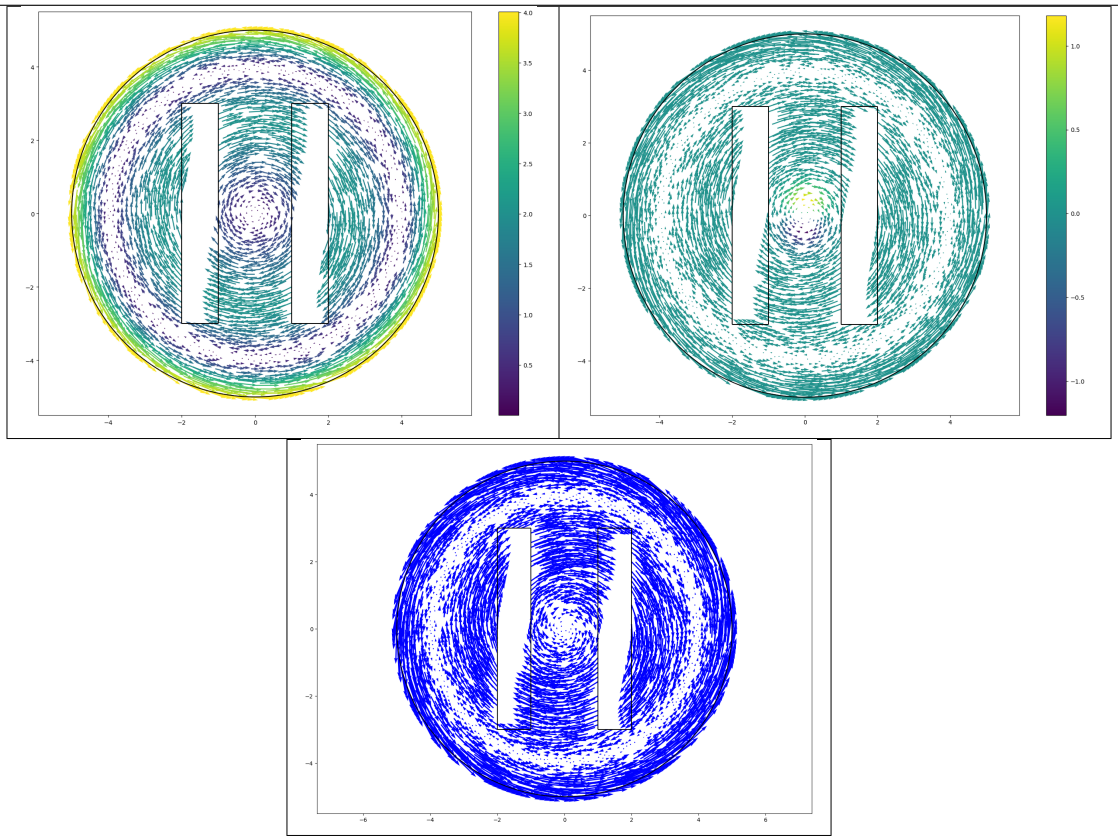
`plt4sim.quiver(q,me,V)` displays vector field `V` on each vertices of the `d`-dimensional simplices elements in dimension $d = 2$ or $d = 3$. The data `V` is an 2D-array numpy array of size *dim*-by-`nq` .

`plt4sim.quiver(q,me,V,Key=Value, ...)` specifies function options using one or more `Key`,`Value` pair arguments. Options of first level are

- `scalars` : to set quivers color to a numpy array of size `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.

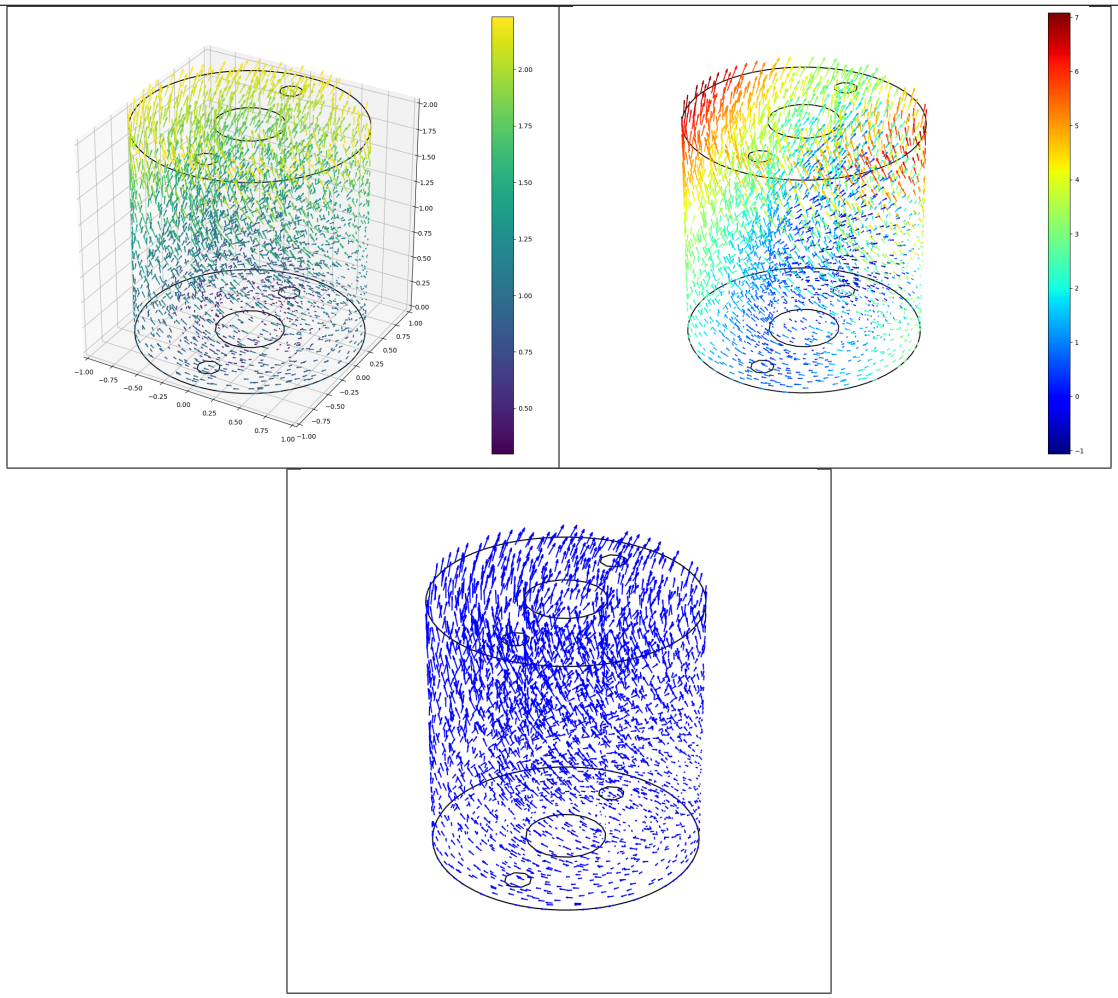
5.1 2D example



```
q,me,toG=mshsim.getMesh2D(2)
q1,me1,toG1=mshsim.getMesh2D(1)
f=lambda x,y:5*np.exp(-3*(x**2+y**2))*np.cos(x)*np.sin(y)
u=f(q[0],q[1])
w=[lambda x,y: y*np.cos(-(x**2+y**2)/10), lambda x,y: -x*np.cos(-(x**2+y**2)/10)]
W=np.array([w[0](q[0],q[1]),w[1](q[0],q[1])])
cmap=plt.cm.get_cmap(name='viridis')
plt.ion()
plt.close('all')
plt.figure(1)
plt4sim.quiver(q,W,scale=50,nvec=3000)
plt4sim.plotmesh(q1,me1,color='Black')
plt.colorbar()
mpltools.set_axes_equal()
plt.figure(2)
plt4sim.quiver(q,W,scalars=u,scale=50,nvec=3000)
plt4sim.plotmesh(q1,me1,color='Black')
plt.colorbar()
mpltools.set_axes_equal()
plt.figure(3)
plt4sim.quiver(q,W,scale=50,nvec=3000,color='Blue')
plt4sim.plotmesh(q1,me1,color='Black')
mpltools.set_axes_equal()
```

Listing 8: 2D mesh : `quiver` function

5.2 3D example

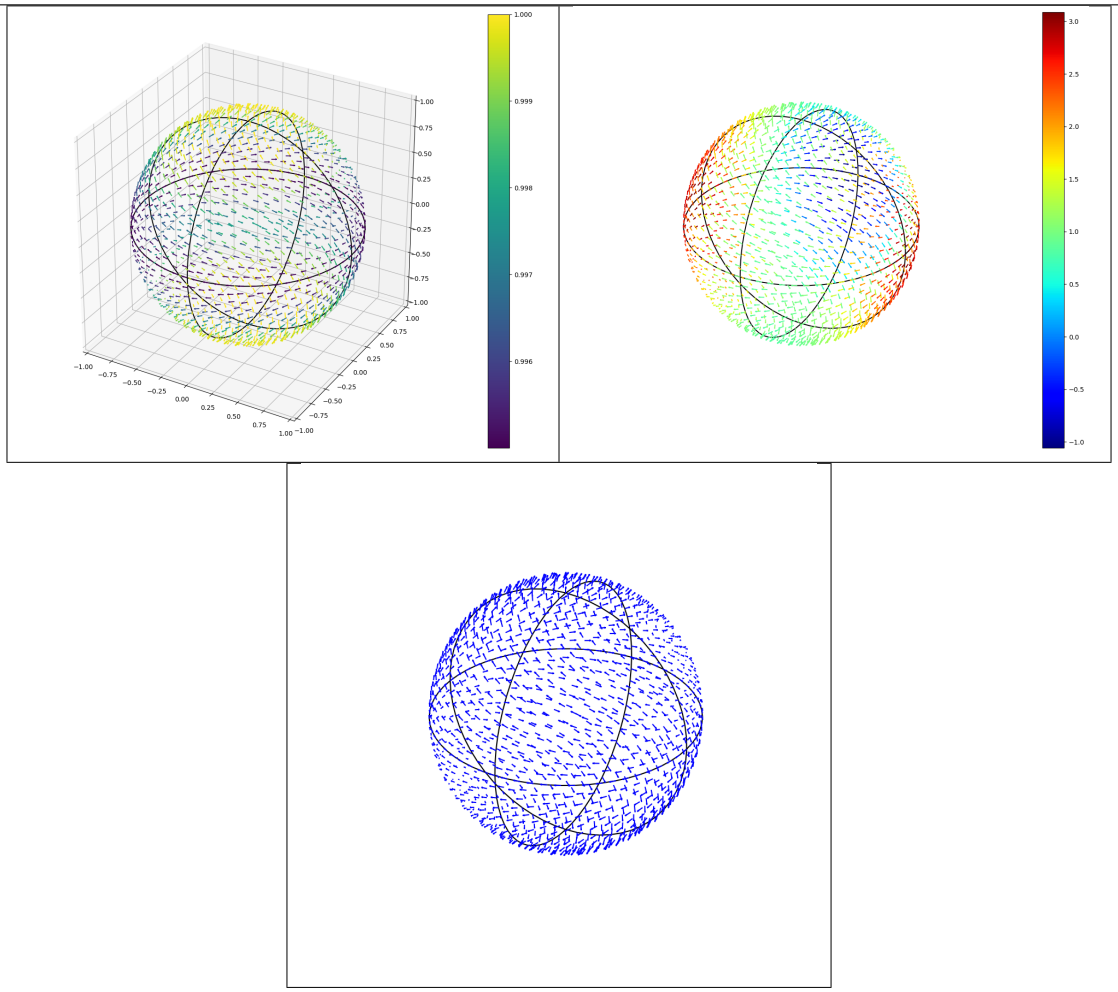


```
q3,me3,toG3=mshsim.getMesh3D(3)
q2,me2,toG2=mshsim.getMesh3D(2)
q1,me1=mshsim.getMesh3D(1)[:2]
f=lambda x,y,z: 3*x**2-y**3+z**2+x*y
w=[lambda x,y,z: y*np.cos(-(x**2+y**2)/10), lambda x,y,z: -x*np.cos(-(x**2+y**2)/10), lambda
    x,y,z: z]
q=q3
u3=f(q[0],q[1],q[2])
W3=np.array([w[0](q[0],q[1],q[2]),w[1](q[0],q[1],q[2]),w[2](q[0],q[1],q[2])])
q=q2
u2=f(q[0],q[1],q[2])
W2=np.array([w[0](q[0],q[1],q[2]),w[1](q[0],q[1],q[2]),w[2](q[0],q[1],q[2])])
q=q1
u1=f(q[0],q[1],q[2])
W1=np.array([w[0](q[0],q[1],q[2]),w[1](q[0],q[1],q[2]),w[2](q[0],q[1],q[2])])
cmap=plt.cm.get_cmap(name='jet')
plt.figure(1)
pq=plt4sim.quiver(q3,W3,scale=20,nvec=3000)
plt.colorbar(pq)
plt4sim.plotmesh(q1,me1,color='Black')
mpltools.set_axes_equal()
plt.figure(2)
pq=plt4sim.quiver(q3,W3,scalars=u3,scale=20,nvec=3000,cmap=cmap)
plt4sim.plotmesh(q1,me1,color='Black')
plt.colorbar(pq)
mpltools.set_axes_equal()
plt.axis('off')
plt.figure(3)
plt4sim.quiver(q3,W3,scale=20,nvec=3000,color='Blue')
plt4sim.plotmesh(q1,me1,color='Black')
mpltools.set_axes_equal()
plt.axis('off')
```

Listing 9: 3D mesh : `quiver` function

5.3 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.



```
q2,me2,toG2=mshsim.getMesh3Ds(2)
q1,me1,toG1=mshsim.getMesh3Ds(1)
f=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]
q=q2
u2=f(q[0],q[1],q[2])
W2=np.array([w[0](q[0],q[1],q[2]),w[1](q[0],q[1],q[2]),w[2](q[0],q[1],q[2])])
q=q1
u1=f(q[0],q[1],q[2])
W1=np.array([w[0](q[0],q[1],q[2]),w[1](q[0],q[1],q[2]),w[2](q[0],q[1],q[2])])
cmap=plt.cm.get_cmap(name='jet')
plt.ion()
plt.close('all')
plt.figure(1)
pq=plt4sim.quiver(q2,W2,scale=20,nvec=3000)
plt.colorbar(pq)
plt4sim.plotmesh(q1,me1,color='Black')
mpltools.set_axes_equal()
plt.figure(2)
pq=plt4sim.quiver(q2,W2,scalars=u2,scale=20,nvec=3000,cmap=cmap)
plt4sim.plotmesh(q1,me1,color='Black')
plt.colorbar(pq)
mpltools.set_axes_equal()
plt.axis('off')
plt.figure(3)
plt4sim.quiver(q2,W2,scale=20,nvec=3000,color='Blue')
plt4sim.plotmesh(q1,me1,color='Black')
mpltools.set_axes_equal()
plt.axis('off')
```

Listing 10: 3D surface mesh : `quiver` function

5 References

[1] F. Cuvelier. fc_mayavi4mesh: a Python package for displaying simplices meshes or datas on simplices meshes by using mayavi python package. `http://www.math.univ-paris13.fr/~cuvelier/software/`, 2017. User's Guide.