

Documentation of the mooMesh Matlab toolbox

*

François Cuvelier[†]

December 5, 2016

Abstract

This object-oriented Matlab toolbox allows to use simplices meshes generated from `gmsh` (in dimension 2 or 3) or an hypercube triangulation (in any dimension). A particular care was taken to the graphics representations of meshes and datas on meshes.

Contents

1	Mesh Objects	3
1.1	Mesh samples	3
1.1.1	2-simplicial mesh in $\mathbb{R}^2$	3
1.1.2	Sample of a 3-simplicial mesh in $\mathbb{R}^3$	4
1.1.3	Sample of a 2-simplicial mesh in $\mathbb{R}^3$	5
1.2	<code>ooMeshElt</code> object	5
1.3	<code>ooMesh</code> object	6
1.4	Partitioned mesh, <code>ooMeshDD</code> object	7
1.4.1	2D samples	9
1.4.2	3D samples	10
1.4.3	3D surface samples	13
2	Data representation on meshes	14
2.1	Data 2D mesh	14
2.1.1	2D partitioned mesh	17
2.2	3D mesh	19
2.2.1	Mapping of the unit ball	22
2.3	3D surface meshes	24
2.3.1	Unit sphere	24
2.3.2	Mapping of the unit sphere	25
2.3.3	3D partitioned mesh	27
2.3.4	3D partitioned surface mesh	29
2.4	Vector field representation on meshes	30

*Compiled with Matlab 2015b

[†]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.

2.4.1	2D mesh	30
2.4.2	2D partitioned mesh	31
2.4.3	3D mesh	32
2.4.4	3D partitioned mesh	33
2.4.5	3D surface mesh	34
2.4.6	3D partitioned surface mesh	35
3	Functions of mooMesh toolbox	36
3.1	gmsh interface	36
3.1.1	gmsh installation (Linux 64bit)	36
3.1.2	Configuration in mooMesh toolbox	36
3.1.3	function gmsh.buildmesh2d	37
3.1.4	function gmsh.buildmesh3d	40
3.1.5	function gmsh.buildmesh3ds	40
3.1.6	function gmsh.buildpartmesh2d	40
3.1.7	function gmsh.buildpartmesh3d	41
3.1.8	function gmsh.buildpartmesh3ds	41
3.1.9	function gmsh.buildPartRectangle	42
3.2	ooMesh methods	43
3.2.1	ooMesh constructor	43
3.2.2	function plot	44
3.2.3	function plotVal	49
3.2.4	function plotIsolines	53
3.2.5	function slicemesh	56
3.2.6	function slice	57
3.2.7	function isoslice	59
3.2.8	function plotVectorField	60
3.3	ooMeshDD methods	63
3.3.1	ooMeshDD constructor	63
3.3.2	function eval	64
3.3.3	function plot	67
3.3.4	function plotPhysicals	71
3.3.5	function plotInterfaces	75
3.3.6	function plotVal	77
3.3.7	function plotIsolines	81
3.3.8	function plotVectorField	85
3.3.9	function slice	90
3.3.10	function slicemesh	91
3.3.11	function isoslice	92
3.3.12	function plotInterfacesVal	93
3.3.13	function plotInterfacesIsolines	97
3.4	ooMesh representations	99
3.5	Datas representations on Meshes	99
A	Calcul des gradients des coordonnées barycentriques	99
B	gmshoptions	101

1 Mesh Objects

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

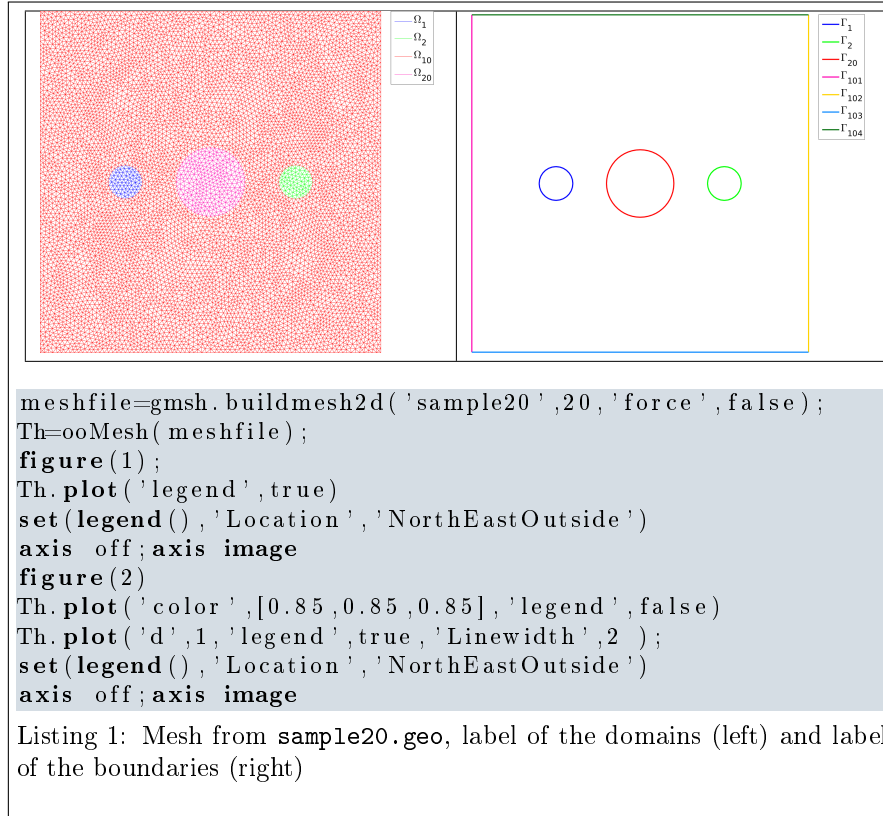
$$K = \left\{ \sum_{i=0}^k \lambda_i q^i \mid \lambda_i \geq 0, i \in \llbracket 0, k \rrbracket, \text{ with } \sum_{i=0}^k \lambda_i = 1 \right\}.$$

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

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

1.1 Mesh samples

1.1.1 2-simplicial mesh in $\mathbb{R}^2$



For example, from figures of Listing 12 the complete domain is

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

and we note

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

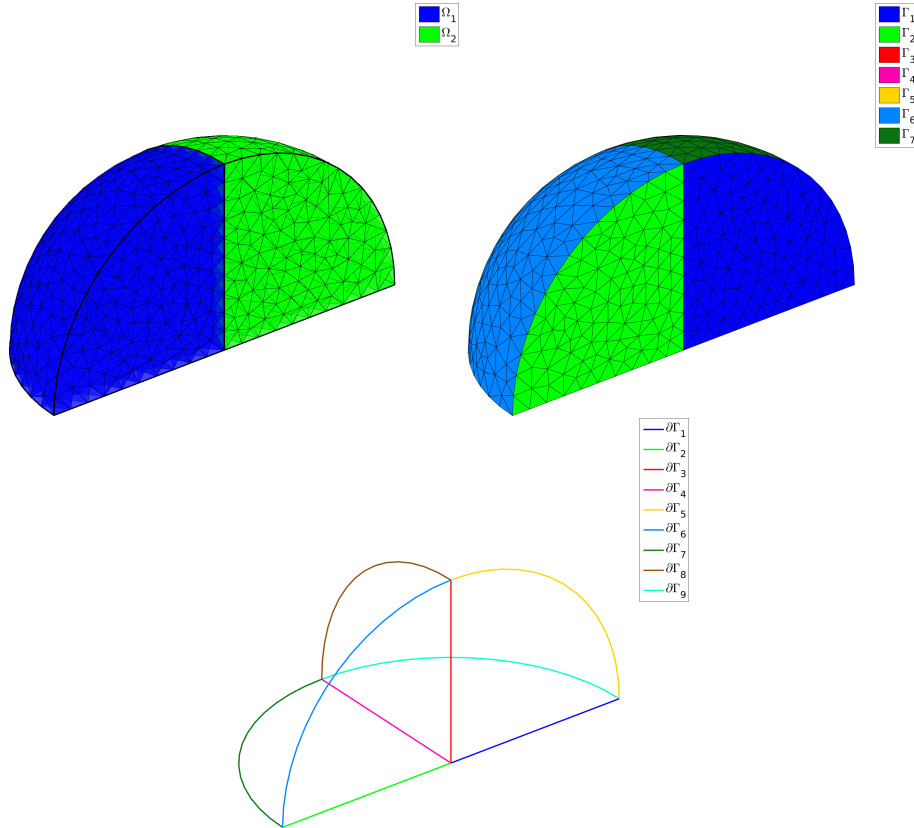
So this mesh is 2-simplicial mesh in $\mathbb{R}^2$ and is composed of :

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

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

```
meshfile=gmsht.buildmesh3d('quart_sphere2',5);
Th=ooMesh(meshfile);
figure(1);
Th.plot('FaceAlpha',0.5,'legend',true);
Th.plot('d',1,'color',[0,0,0],'LineWidth',2);
axis off;axis image;
figure(2);
Th.plot('d',2,'legend',true);
axis off;axis image;
figure(3);
Th.plot('d',1,'LineWidth',2,'legend',true);
axis off;axis image
```

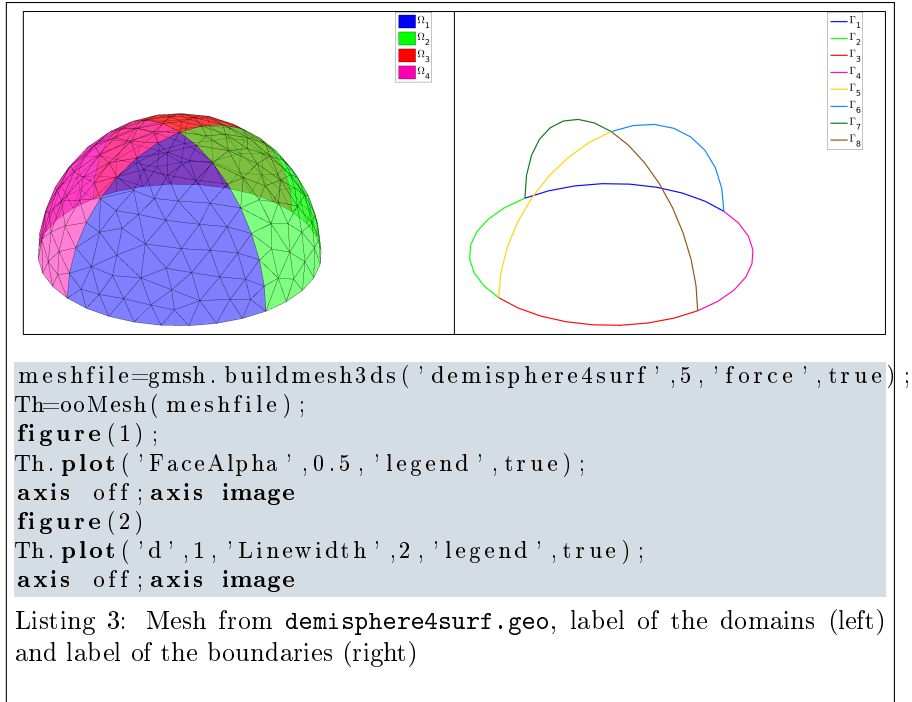
Listing 2: Mesh from quart_sphere2.geo



The mesh obtained from Listing 13 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 : $\partial\Gamma_i \forall i \in \llbracket 1, 9 \rrbracket$

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



The mesh obtained from Listing 9 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 : $\Gamma_i \forall i \in \llbracket 1, 8 \rrbracket$

1.2 ooMeshElt object

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



Properties of ooMeshElt object for d-simplicial elementary meshes in $\mathbb{R}^{\dim}$

dim	: integer space dimension
d	: integer ($0 \leq d \leq \dim$)
n _q	: integer number of vertices
n _{me}	: integer number of elements (d-simplices)
q	: dim-by-n _q array of reals array of vertex coordinates
me	: (d + 1)-by-n _{me} array of integers connectivity array for mesh elements
vols	: 1-by-n _{me} array of reals array of mesh element volumes
h	: double mesh step size (=maximum edge length in the mesh)
toGlobal	: 1-by-n _q array of integers convert from local to global mesh vertices numbering
toParent	: 1-by-n _q array of integers convert from local to parent mesh vertices numbering (same as global if not part of a partitioned mesh)

More precisely

- $q(\nu, j)$ is the ν -th coordinate of the j -th vertex, $\nu \in \{1, \dots, \dim\}$, $j \in \{1, \dots, n_q\}$. The j -th vertex will be also denoted by $q^j = q(:, j)$.
- $me(\beta, k)$ is the storage index of the β -th vertex of the k -th element (d-simplex), in the array q , for $\beta \in \{1, \dots, d + 1\}$ and $k \in \{1, \dots, n_{me}\}$. So $q(:, me(\beta, k))$ represents the coordinates of the β -th vertex of the k -th mesh element.
- $vols(k)$ is the volume of the k -th d-simplex .

1.3 ooMesh object

A d-simplicial mesh in dimension $\dim$, represented as an **ooMesh** object, is an union of **ooMeshElt** objects which are elementary l -simplicial meshes ($l \leq d$) in space dimension $\dim$.



Mesh structure associated to $\mathcal{T}_h$

dim	:	integer space dimension
d	:	integer d-dimensional simplicial mesh
sTh	:	array of ooMeshElt objects
nsTh	:	number of ooMeshElt objects
sThsimp	:	array of nsTh integers <i>i</i> -th ooMeshElt object in sTh is a sThsimp(i) - simplicial elementary mesh
sThlab	:	array of nsTh integers in sTh label of <i>i</i> -th ooMeshElt object in sTh is number sThlab(i)
n_q	:	integer number of vertices in $\mathcal{T}_h$
toGlobal	:	1-by- n_q array of integers convert from local to global mesh vertices numbering
toParent	:	1-by- n_q array of integers convert from local to parent mesh vertices numbering (same as global if not part of a partitioned mesh)

1.4 Partitioned mesh, **ooMeshDD** object

A d-simplicial partitioned mesh in dimension **dim**, represented as an **ooMeshDD** object, is an union of **ooMesh** objects which are d-simplicial meshes in space dimension **dim**.

**Partitioned mesh, ooMeshDD object**

dim	:	integer space dimension
d	:	integer d-dimensional simplicial mesh
$\mathcal{T}_h$	:	array of ooMesh objects, one by partition we set $\Omega_i = \mathcal{T}_h(i)$
np	:	number of partitions (i.e. length of $\mathcal{T}_h$)
PhysElts	:	...
InterElts	:	...

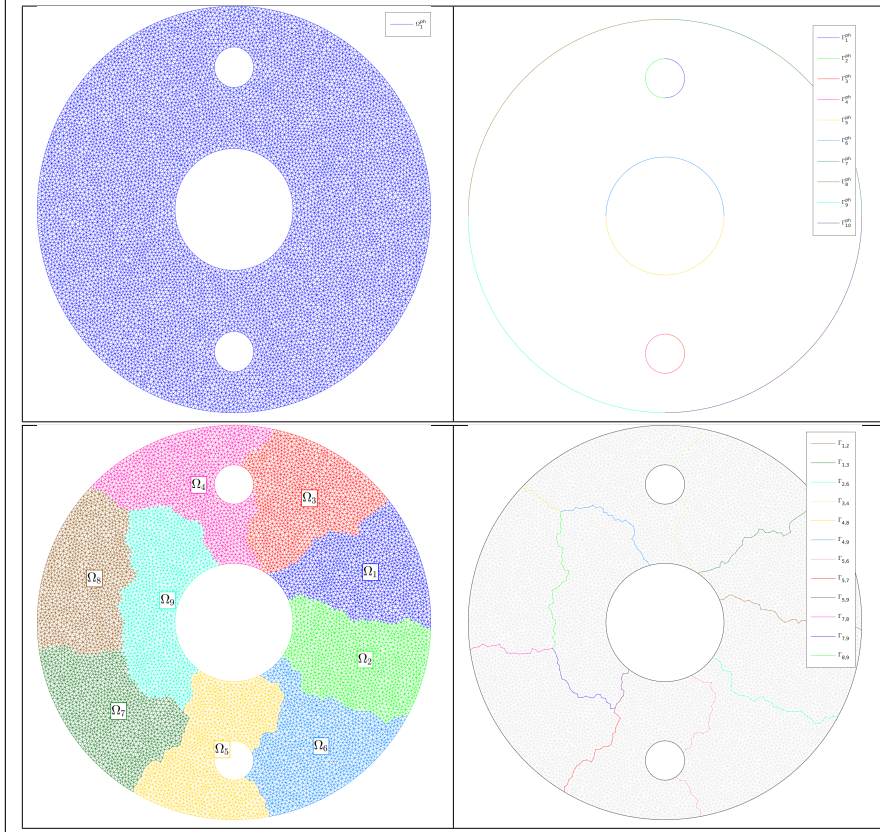
**InterElts, interface element structure associated with an ooMeshDD object**

d	:	integer d-dimensional simplicial interfaces
labels	:	list of interface labels in ooMeshDD object
parts	:	array of np lists parts(i) contains all interface labels of $\Omega_i = \mathcal{T}_h(i)$
Gamma	:	np-by-np sparse matrix. Gamma(i,j) is the interface label of $\partial\Omega_i \cap \partial\Omega_j$ if not empty, 0 otherwise. We set $\Gamma_{i,j} = \text{Gamma}(i,j)$

**PhysElts, physical element structure associated with an ooMeshDD object**

d	:	integer d-dimensional simplicial physical elements
labels	:	list of nl physical labels in ooMeshDD object
parts	:	array of np lists parts(i) contains all physical labels of $\Omega_i = \mathcal{T}_h(i)$
connect	:	nl-by-np sparse matrix. connect(l,i) $\neq$ 0 if labels(l) physical element is partially in Ω_i

1.4.1 2D samples



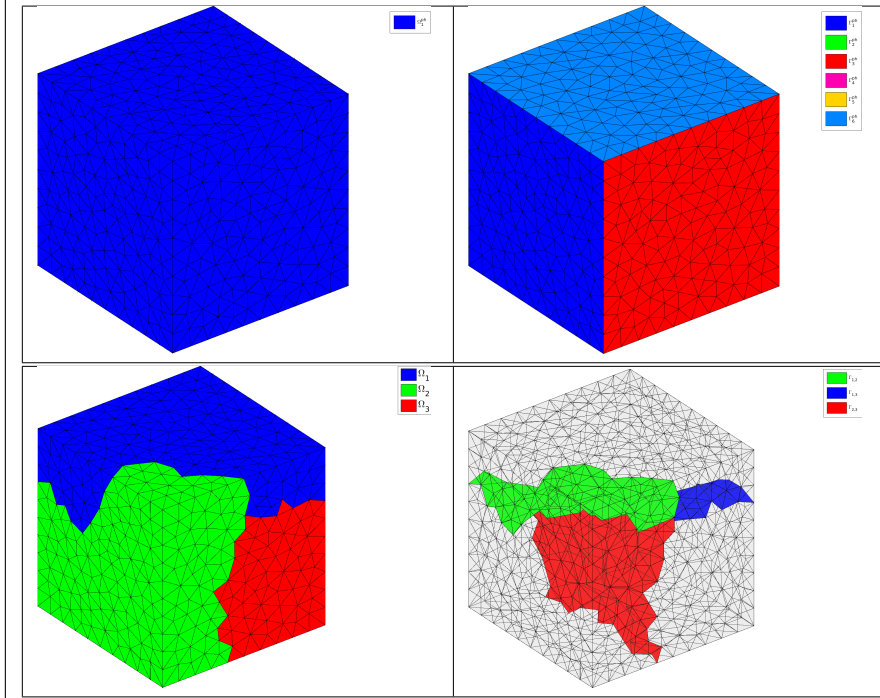
```

meshfile=gmsht.buildmesh2d('disk3holes',50);
meshpartfile=gmsht.buildpartmesh2d(meshfile,9);
ThDD=ooMeshDD(meshpartfile);
figure(1)
ThDD.plotPhysicals('legend',true)
figure(2)
ThDD.plotPhysicals('d',1,'legend',true)
figure(3)
ThDD.plot('legend',true)
figure(4)
ThDD.plot('color',0.9*[1,1,1])
ThDD.plotPhysicals('d',1,'color','k')
ThDD.plotInterfaces('legend',true)

```

Listing 4: Mesh from `disk3holes.geo`, label of the domains (upper left), label of the boundaries (upper right), label of the partitions (bottom left) and label of the interfaces (bottom right)

1.4.2 3D samples

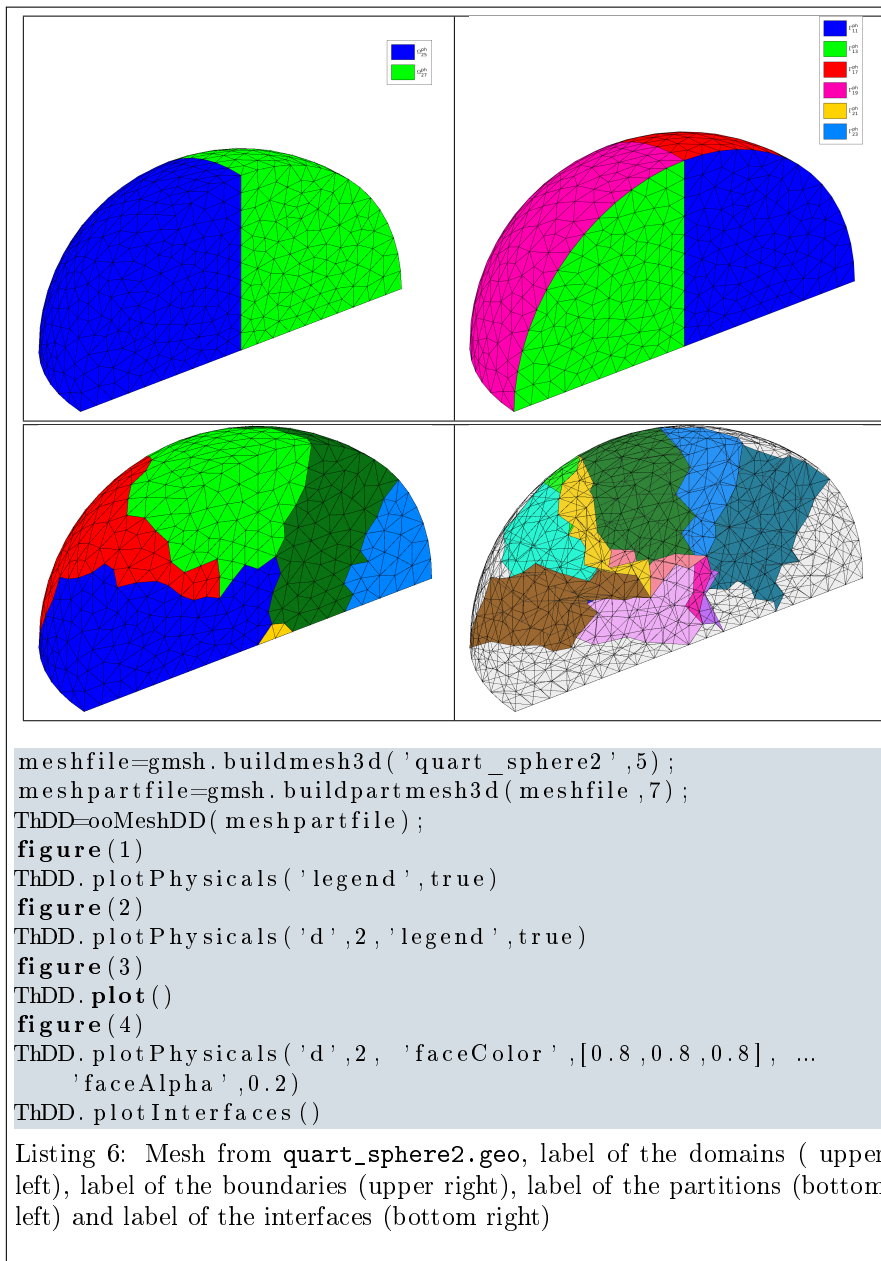


```

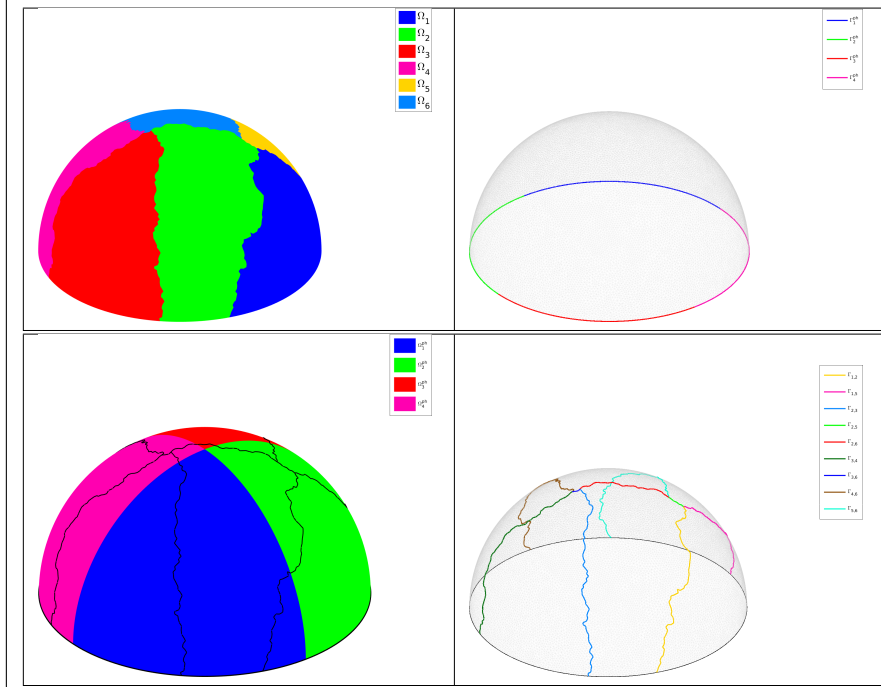
meshfile=gmsb.buildmesh3d('cube6',10);
meshpartfile=gmsb.buildpartmesh3d(meshfile,3);
ThDD=ooMeshDD(meshpartfile);
figure(1)
ThDD.plotPhysicals('legend',true)
figure(2)
ThDD.plotPhysicals('d',2,'legend',true)
figure(3)
ThDD.plot('legend',true)
figure(4)
ThDD.plotPhysicals('d',2,'faceColor',[0.8,0.8,0.8],'faceAlpha',0.2)
ThDD.plotInterfaces('legend',true)

```

Listing 5: Mesh from `cube6.geo`, label of the domains (upper left), label of the boundaries (upper right), label of the partitions (bottom left) and label of the interfaces (bottom right)



1.4.3 3D surface samples



```

meshfile=gmesh.buildmesh3ds('demisphere4surf',50,'force',false);
meshpartfile=gmesh.buildpartmesh3ds(meshfile,6,'force',false);
Th=ooMeshDD(meshpartfile);
figure(1);
Th.plot('EdgeColor','none','legend',true);
axis off;axis image

figure(2)
Th.plot('FaceColor','None',...
        'EdgeColor',0.8*[1,1,1],'EdgeAlpha',0.3);
Th.plotPhysicals('d',1,'LineWidth',2,'legend',true);
axis off;axis image

figure(3)
Th.plotPhysicals('EdgeColor','None','legend',true);
Th.plotPhysicals('d',1,'color',[0,0,0],'LineWidth',2);
Th.plotInterfaces('Color',[0,0,0],'LineWidth',1.5,...
        'LineStyle','-');
axis off;axis image

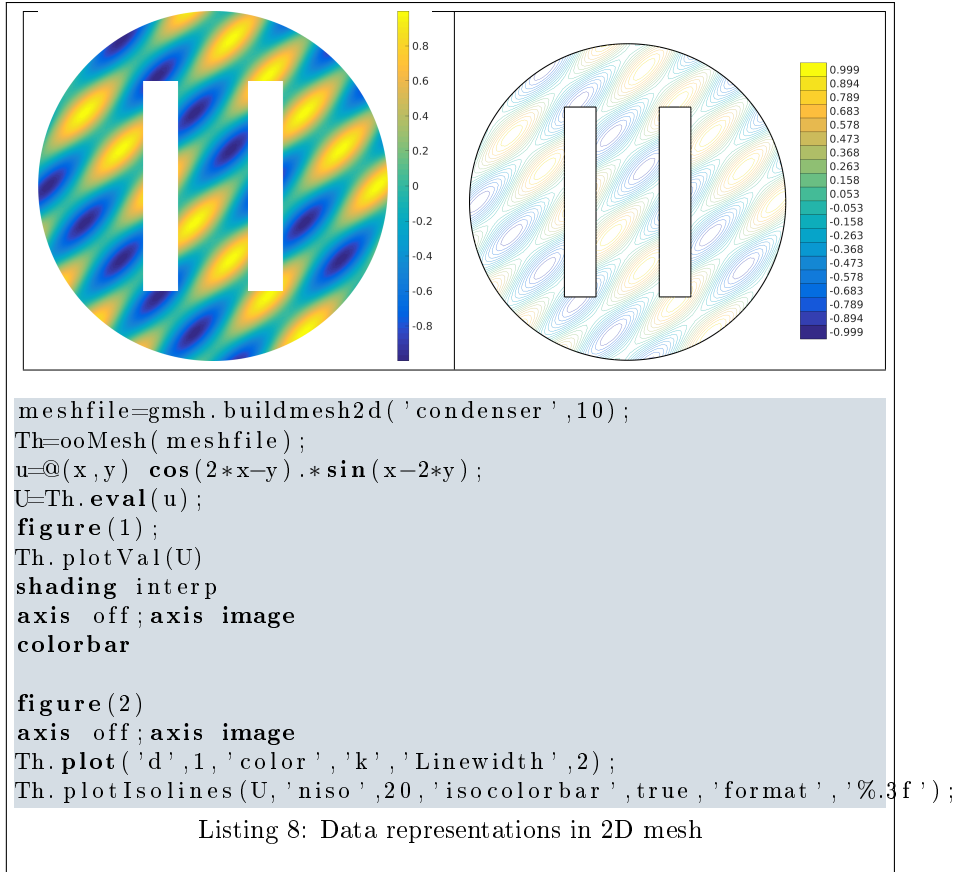
figure(4)
Th.plot('FaceColor','None','EdgeColor',0.8*[1,1,1],...
        'EdgeAlpha',0.3);
Th.plotPhysicals('d',1,'color','k','LineWidth',1);
Th.plotInterfaces('LineWidth',2,'legend',true);
axis off;axis image

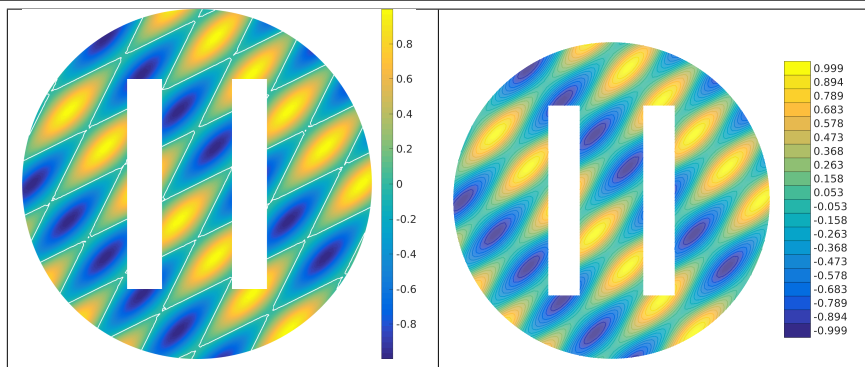
```

Listing 7: Mesh from `demisphere4surf.geo`, label of the domains (upper left), label of the boundaries (upper right), label of the partitions (bottom left) and label of the interfaces (bottom right)

2 Data representation on meshes

2.1 Data 2D mesh





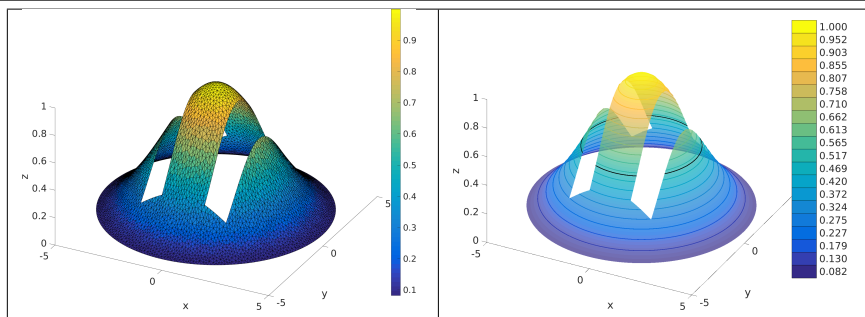
```

meshfile=gmesh.buildmesh2d( 'condenser ',10);
Th=ooMesh( meshfile );
u=@(x,y) cos(2*x-y) .* sin(x-2*y);
U=Th.eval(u);
figure(1)
Th.plotVal(U)
Th.plotIsolines(U, 'isorange',0, 'color','w', ...
    'Linewidth',1.5);
axis off,axis image,shading interp,colorbar

figure(2)
Th.plotVal(U, 'FaceAlpha',0.8)
[Colors, isorange]=Th.plotIsolines(U, 'niso',20, ...
    'Linewidth',1.5);
axis off,axis image, shading interp
colorbarIso(Colors, isorange, '%.3f')

```

Listing 9: Data representations on 2D mesh. White lines are zero isolines (left)



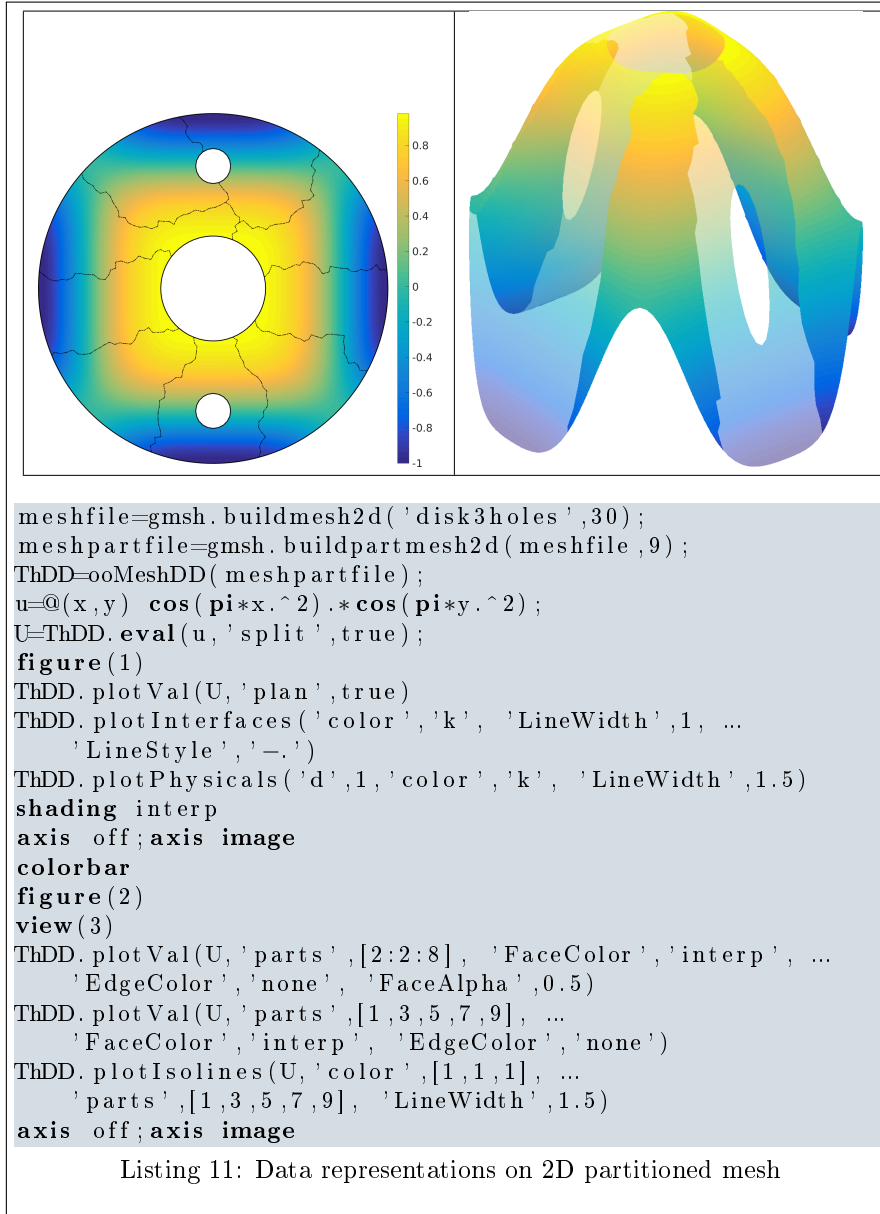
```

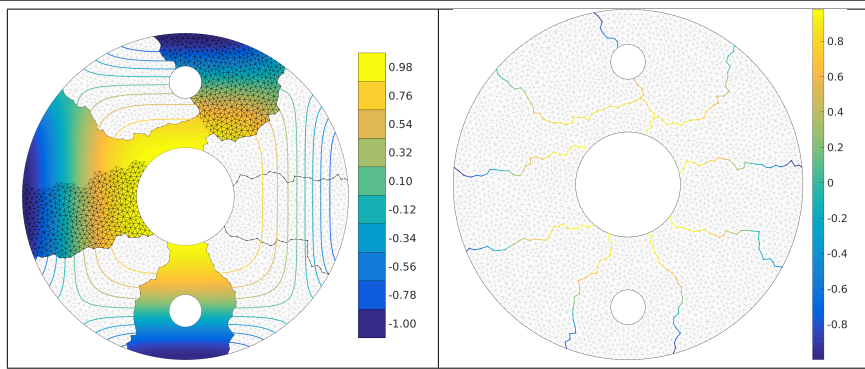
meshfile=gmesh.buildmesh2d('condenser',7);
Th=ooMesh(meshfile);
u=@(x,y)exp(-(x.^2+y.^2)/10);
U=Th.eval(u);
figure(1)
Th.plotVal(U)
colorbar
view(27,39)
xlabel('x'),ylabel('y'),zlabel('z')
figure(2)
Th.plotVal(U,'FaceAlpha',0.7)
[Colors,isorange]=Th.plotIsolines(U,'niso',20,...
    'Linewidth',1.5);
Th.plotIsolines(U,'isorange',0.5,'color','k',...
    'Linewidth',1.5);
colorbarIso(Colors,isorange,'%3f')
shading interp
view(27,39)
xlabel('x'),ylabel('y'),zlabel('z')

```

Listing 10: Data representations on 2D mesh. Black lines are zero isolines (right)

2.1.1 2D partitioned mesh





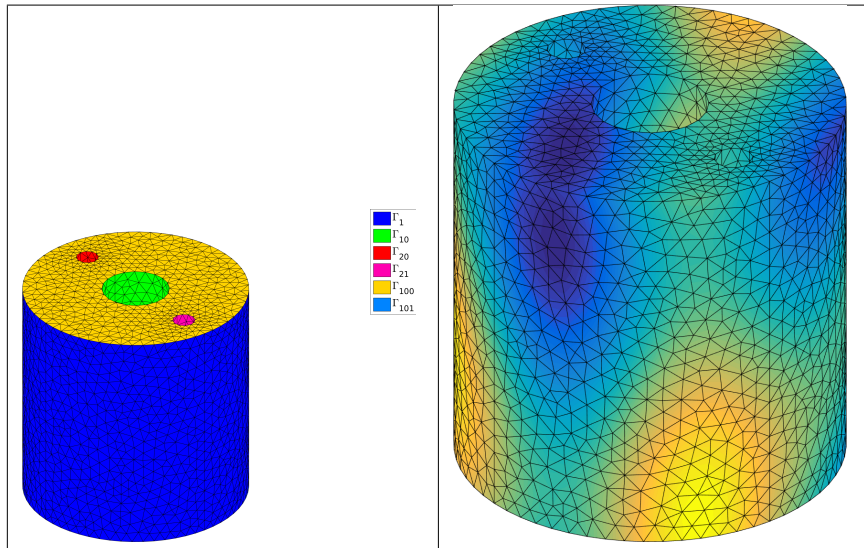
```

meshfile=gmesh.buildmesh2d('disk3holes',30);
meshpartfile=gmesh.buildpartmesh2d(meshfile,9);
ThDD=ooMeshDD(meshpartfile);
u=@(x,y) cos(pi*x.^2).*cos(pi*y.^2);
U=ThDD.eval(u,'split',true);
figure(1)
ThDD.plot('labels',[1,3,5,7,9], 'color',[0.8,0.8,0.8])
ThDD.plotPhysicals('d',1, 'color','k')
ThDD.plotInterfaces('color','k', 'LineWidth',1, ...
    'LineStyle','-')
ThDD.plotVal(U,'plan',true,'parts',[2,6], ...
    'FaceColor','interp')
ThDD.plotVal(U,'plan',true,'parts',[4,8], ...
    'FaceColor','interp', 'EdgeColor','none')
ThDD.plotIsolines(U,'plan',true,'parts',[1,3,5,7,9], ...
    'LineWidth',1.5, 'isocolorbar',true, 'format','%2f')
axis off;axis image

figure(2)
sgu=ThDD.trace(U,'interface',true);
ThDD.plot('color',[0.8,0.8,0.8])
ThDD.plotPhysicals('d',1, 'color','k')
ThDD.plotInterfacesVal(sgu,'plan',true, 'LineWidth',1.5)
axis off;axis image
colorbar

```

Listing 12: Data representations on 2D partitioned mesh

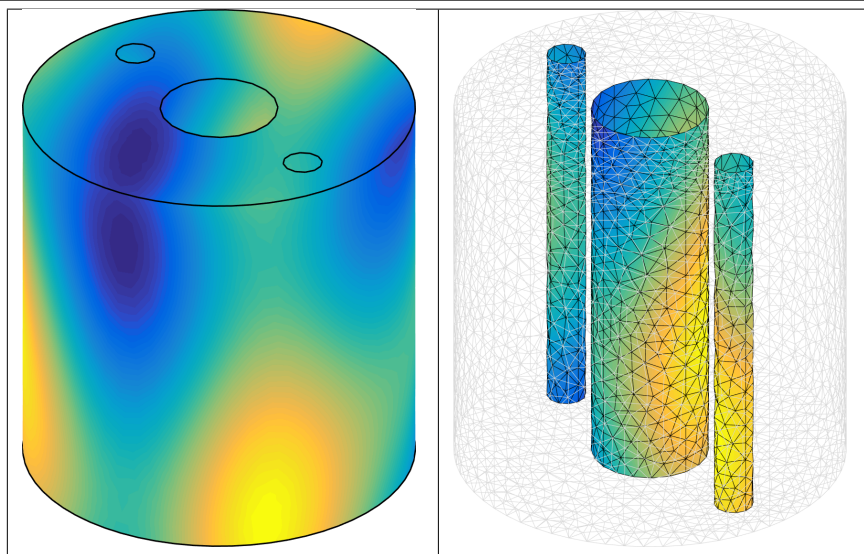


```

meshfile=gmesh.buildmesh3d('cylinder3holes',10);
Th=ooMesh( meshfile );
u=@(x,y,z) cos(2*x-y-z) .* sin(x-2*y+z);
U=Th.eval(u);
figure(1);
Th.plot('d',2,'legend',true)
axis off;axis image
figure(2)
axis off;axis image
Th.plotVal(U,'d',2)

```

Listing 13: Data representations on 3D mesh

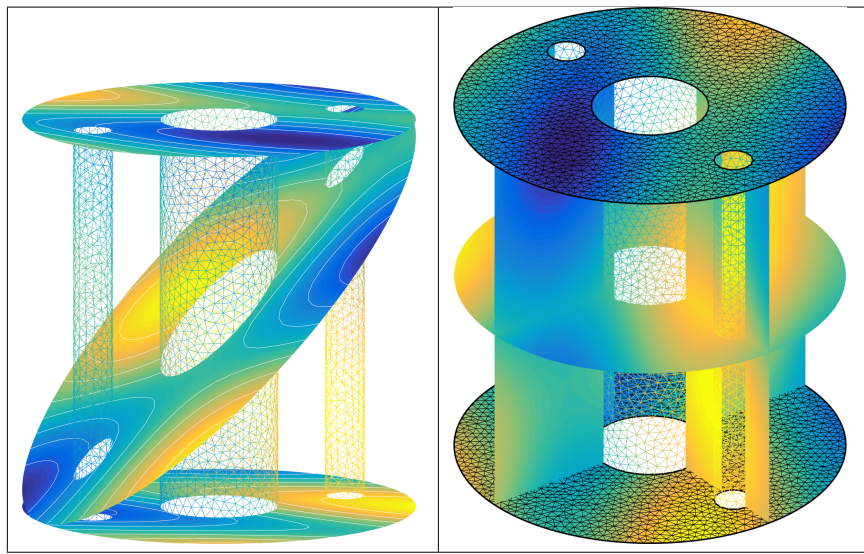


```

meshfile=gmsl.buildmesh3d('cylinder3holes',10);
Th=ooMesh(meshfile);
u=@(x,y,z) cos(2*x-y-z).*sin(x-2*y+z);
U=Th.eval(u);
figure(1)
axis off;axis image
Th.plotVal(U,'d',2)
caxis([min(U),max(U)])
shading interp
Th.plot('d',1,'color','k','Linewidth',1.5)
figure(2)
axis off;axis image
Th.plot('d',2,'FaceColor','none',...
'EdgeColor',0.85*[1,1,1])
Th.plotVal(U,'d',2,'labels',[10,20,21])
caxis([min(U),max(U)])

```

Listing 14: Data representations on 3D mesh



```

meshfile=gmsb.buildmesh3d('cylinder3holes',20);
Th=ooMesh(meshfile);
u=@(x,y,z) cos(2*x-y-z).*sin(x-2*y+z);
U=Th.eval(u);
figure(1)
P=PlaneCoefs([0 0 1],[0 1 1]);
Th.slice(U,P,'FaceColor','interp','EdgeColor','none')
axis off,axis image,hold on
Th.plotVal(U,'d',2,'labels',[100,101], ...
'FaceColor','interp','EdgeColor','none')
caxis([min(U),max(U)])
isorange=linspace(min(U),max(U),10);
Th.plotVal(U,'d',2,'labels',[10,20,21], ...
'FaceColor','none','EdgeColor','interp')
Th.plotIsolines(U,'labels',[100,101], ...
'isorange',isorange,'color','w')
Th.isoslice(U,P,'isorange',isorange,'color','w')
view(-114,11)
figure(2)
Th.slice(U,PlaneCoefs([0 0 1],[1 0 0]), ...
'FaceColor','interp','EdgeColor','none')
axis off,axis image,hold on
Th.slice(U,PlaneCoefs([0 0 1],[0 1 0]), ...
'FaceColor','interp','EdgeColor','none')
Th.slice(U,PlaneCoefs([0 0 1],[0 0 1]), ...
'FaceColor','interp','EdgeColor','none')
Th.plotVal(U,'d',2,'labels',[100,101], ...
'FaceColor','interp')
Th.plotVal(U,'d',2,'labels',[10,20,21], ...
'FaceColor','none','EdgeColor','interp')
caxis([min(U),max(U)])
Th.plot('d',1,'Color','k','Linewidth',1.5)

```

Listing 15: Data representations on 3D mesh

2.2.1 Mapping of the unit ball

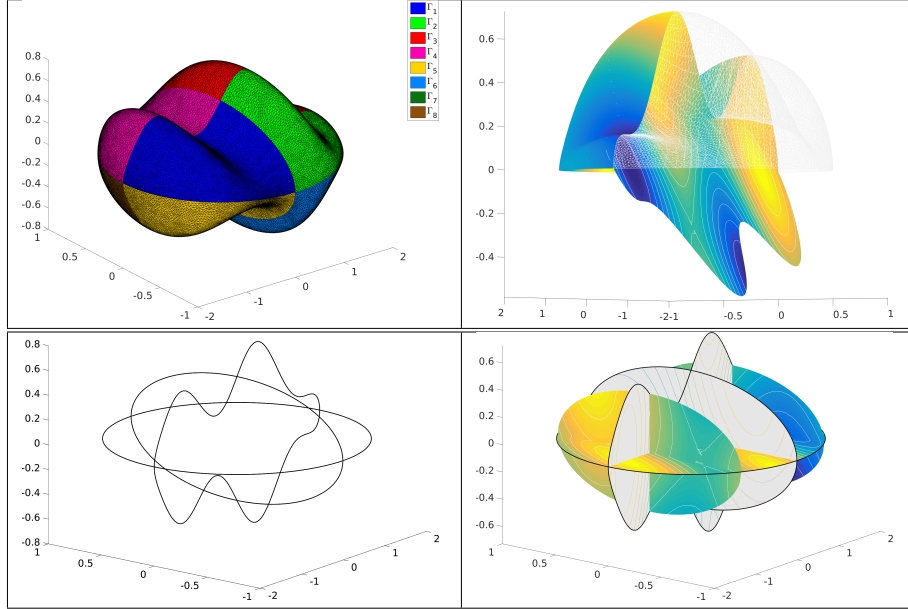
The 3D domain Ω is constructed by mapping a discretization of the unit ball onto Ω by

$$F(y_1, y_2, y_3) = \begin{pmatrix} 2y_1 \\ y_2 \\ \frac{1}{2}y_3(1 + \frac{1}{2}\sin(2\pi y_1)) \end{pmatrix}, \quad \forall \mathbf{y} = (y_1, y_2, y_3), \text{ such that } \|\mathbf{y}\|_2 \leq 1$$

We represent the function

$$u(x, y, z) = \cos(2x - y - z) \sin(x - 2y + z)$$

on Ω by mapping the unit ball obtained from `gmsh` with `ball18.geo`.



```

trans=@(q) [2*q(1,:);q(2,:);0.5*q(3,:).*(1+0.5*sin(2*pi*q(1,:)))];
meshfile=gmesh.buildmesh3d('ball8',40);
Th=ooMesh(meshfile,'trans',trans);
u=@(x,y,z) cos(2*x-y-z).*sin(x-2*y+z);
U=Th.eval(u);
figure(1)
axis off;axis image
Th.plot('d',2,'legend',true);
Options={'FaceColor','interp','EdgeColor','none'};
figure(2)
axis off;axis image
P=PlaneCoefs([0 0 0],[0 1 1]);
Th.slice(U,P)
hold on
Th.plotVal(U,'d',2,'labels',[1,2],'EdgeColor','None')
caxis([min(U),max(U)])
isorange=linspace(min(U),max(U),15);
Th.plot('d',2,'labels',[3,4],'FaceColor','none',...
'EdgeColor',0.95*[1,1,1])
Th.plotIsolines(U,'labels',[3,4],'isorange',isorange,'color','w')
Th.isoslice(U,P,'isorange',isorange,'color','w')
view(127,-1)
figure(3)
axis off;axis image
P1=PlaneCoefs([0 0 0],[1 0 0]);
Th.slice(U,P1,Options{:})
Th.isoslice(U,P1,'isorange',isorange,'color','w')
P2=PlaneCoefs([0 0 0],[0 1 0]);
Th.slice(U,P2,Options{:})
Th.isoslice(U,P2,'isorange',isorange,'color','w')
P3=PlaneCoefs([0 0 0],[0 0 1]);
Th.slice(U,P3,Options{:})
Th.isoslice(U,P3,'isorange',isorange,'color','w')
P4=PlaneCoefs([-1 0 0],[1 0 0]);
Th.slice(U,P4,Options{:})
Th.isoslice(U,P4,'isorange',isorange,'color','w')
P5=PlaneCoefs([1 0 0],[1 0 0]);
Th.slice(U,P5,Options{:})
Th.isoslice(U,P5,'isorange',isorange,'color','w')
Th.plot('d',1,'color','k','Linewidth',1.5)
view(-52,20)
figure(4)
axis off;axis image
P1=PlaneCoefs([0 0 0],[1 0 0]);
Th.slicemesh(P1,'FaceColor',0.9*[1 1 1],'EdgeColor','none')
hold on
Th.isoslice(U,P1,'isorange',isorange)
P2=PlaneCoefs([0 0 0],[0 1 0]);
Th.slicemesh(P2,'FaceColor',0.9*[1 1 1],'EdgeColor','none')
Th.isoslice(U,P2,'isorange',isorange)
P3=PlaneCoefs([0 0 0],[0 0 1]);
Th.slice(U,P3,'FaceColor','interp')
Th.isoslice(U,P3,'isorange',isorange,'color','w')
P4=PlaneCoefs([-1 0 0],[1 0 0]);
Th.slice(U,P4,'FaceColor','interp')
Th.isoslice(U,P4,'isorange',isorange,'color','w')
P5=PlaneCoefs([1 0 0],[1 0 0]);
Th.slice(U,P5,'FaceColor','interp')
Th.isoslice(U,P5,'isorange',isorange,'color','w')
Th.plot('d',1,'legend',false,'color','k','Linewidth',1.5)
view(-52,20)

```

Listing 16: Data representations on 3D mesh

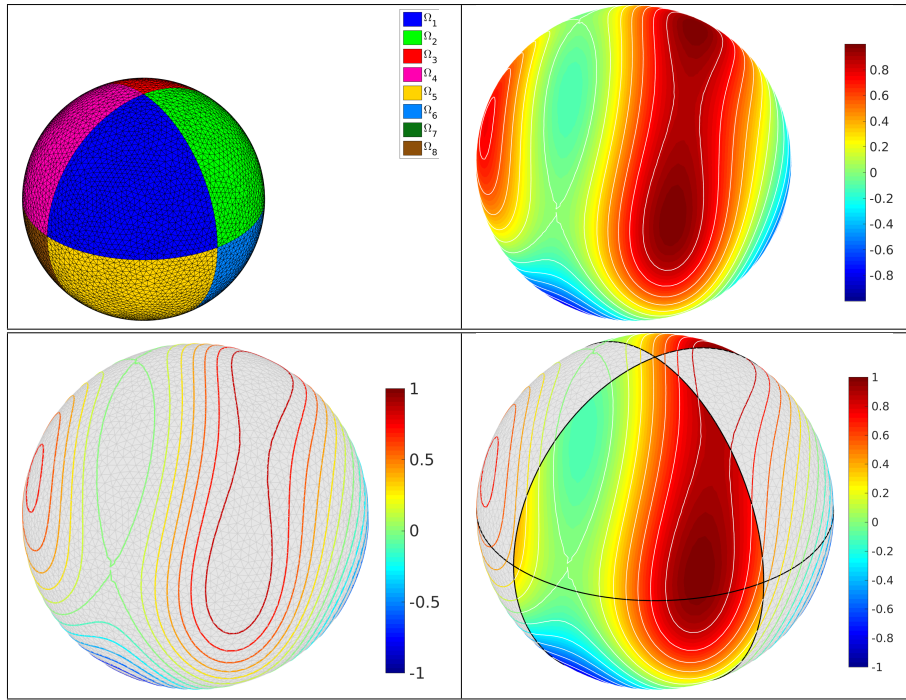
2.3 3D surface meshes

2.3.1 Unit sphere

We represent the function

$$u(x, y, z) = \cos(2x - y - z) \sin(x - 2y + z)$$

on the unit sphere obtained from `gmsh` with `sphere8surf.geo`.



```
meshfile=gmsh.buildmesh3ds('sphere8surf',20);
Th=ooMesh(meshfile);
u=@(x,y,z) cos(2*x-y-z).*sin(x-2*y+z);
U=Th.eval(u);
figure(1);
Th.plot('legend',true)
axis off;axis image
hfig=figure(2);
Th.plotVal(U,'d',2)
Th.plotIsolines(U,'niso',15,'color',[1 1 1], 'Linewidth',1)
caxis([min(U),max(U)])
axis off, axis image
shading interp, colormap(jet)
h=colorbar;
set(hfig,'Position',[0 0 800 600])
set(h,'Position',[0.75 0.25 0.025 0.50])
figure(3)
colormap(jet)
Th.plot('FaceColor',0.9*[1 1 1], 'EdgeColor',0.8*[1 1 1]);
```

```

Th.plotIsolines(U, 'niso', 15, 'Linewidth', 1.5)
axis off, axis image
h=colorbar;
set(hfig, 'Position', [0 0 800 600])
set(h, 'Position', [0.75 0.25 0.025 0.50])
hfig=figure(4);
colormap(jet)
Th.plotVal(U, 'd', 2, 'labels', [1:2:8])
Th.plotIsolines(U, 'labels', [1:2:8], 'niso', 15, 'color', [1 1 ...
    1], 'Linewidth', 1)
shading interp
Th.plot('labels', [2:2:8], 'FaceColor', 0.9*[1 1 1], ...
    'EdgeColor', 0.8*[1 1 1]);
Th.plotIsolines(U, 'labels', [2:2:8], 'niso', 15, 'Linewidth', 1.5)
axis off, axis image
Th.plot('d', 1, 'color', [0 0 0], 'Linewidth', 1.5)
h=colorbar;
set(hfig, 'Position', [0 0 800 600])
set(h, 'Position', [0.75 0.25 0.025 0.50])

```

Listing 17: Data representations on 3D surface mesh

2.3.2 Mapping of the unit sphere

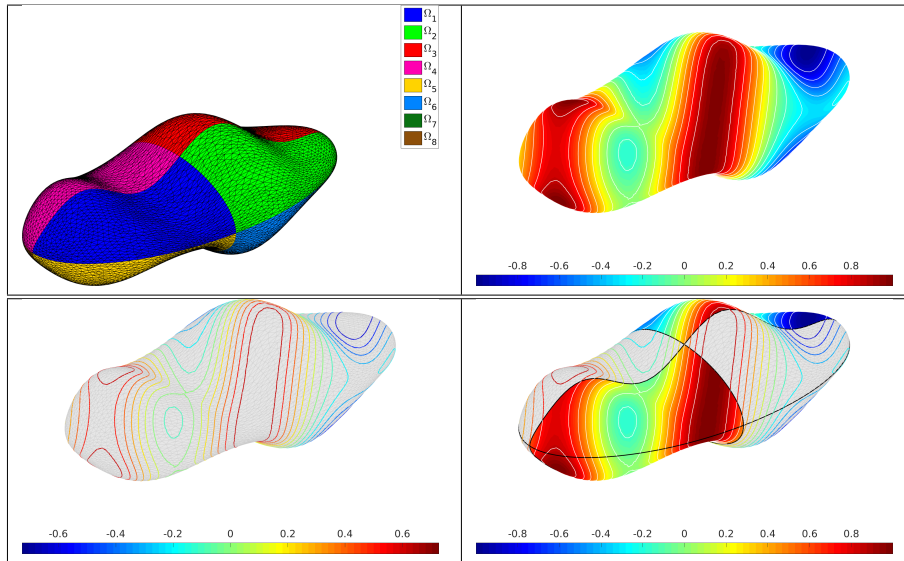
The hypersurface is constructed by mapping a discretization of the unit sphere S^2 onto the surface Ω by

$$F(y_1, y_2, y_3) = \begin{pmatrix} 2y_1 \\ y_2 \\ \frac{1}{2}y_3(1 + \frac{1}{2}\sin(2\pi y_1)) \end{pmatrix}, \quad \forall \mathbf{y} = (y_1, y_2, y_3) \in S^2$$

We represent the function

$$u(x, y, z) = \cos(2x - y - z) \sin(x - 2y + z)$$

on the surface Ω by mapping the unit sphere obtained from gmsh with `sphere8surf.geo`.



```

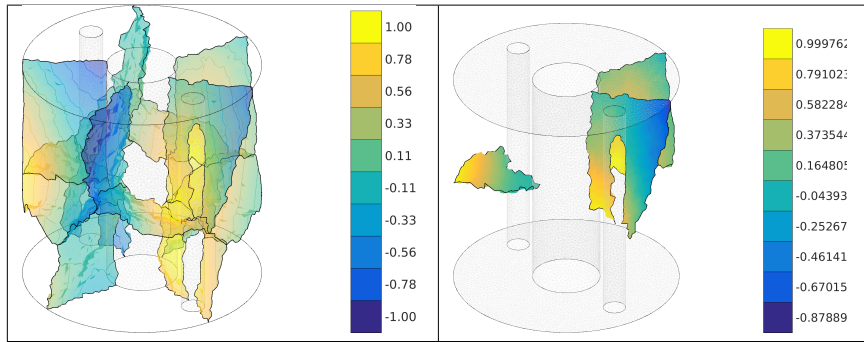
meshfile=gmesh.buildmesh3ds('sphere8surf',20);
Th=ooMesh(meshfile);
u=@(x,y,z) cos(2*x-y-z).*sin(x-2*y+z);
U=Th.eval(u);
figure(1);
Th.plot('legend',true)
axis off;axis image
hfig=figure(2);
Th.plotVal(U,'d',2)
Th.plotIsolines(U,'niso',15,'color',[1 1 1], 'Linewidth',1)
caxis([min(U),max(U)])
axis off, axis image
shading interp, colormap(jet)
h=colorbar;
set(hfig,'Position',[0 0 800 600])
set(h,'Position',[0.75 0.25 0.025 0.50])
figure(3)
colormap(jet)
Th.plot('FaceColor',0.9*[1 1 1], 'EdgeColor',0.8*[1 1 1]);
Th.plotIsolines(U,'niso',15, 'Linewidth',1.5)
axis off, axis image
h=colorbar;
set(hfig,'Position',[0 0 800 600])
set(h,'Position',[0.75 0.25 0.025 0.50])
hfig=figure(4);
colormap(jet)
Th.plotVal(U,'d',2,'labels',[1:2:8])
Th.plotIsolines(U,'labels',[1:2:8], 'niso',15,'color',[1 1 1], ...
'Linewidth',1)
shading interp
Th.plot('labels',[2:2:8], 'FaceColor',0.9*[1 1 1], 'EdgeColor',0.8*[1 1 ...
1]);
Th.plotIsolines(U,'labels',[2:2:8], 'niso',15, 'Linewidth',1.5)
axis off, axis image
Th.plot('d',1,'color',[0 0 0], 'Linewidth',1.5)
h=colorbar;
set(hfig,'Position',[0 0 800 600])
set(h,'Position',[0.75 0.25 0.025 0.50])

```

Listing 18: Data representations on 3D surface mesh

2.3.3 3D partitioned mesh





```

meshfile=gmsht.buildmesh3d('cylinder3holes',30);
Th=ooMesh(meshfile);
meshpartfile=gmsht.buildpartmesh3d(meshfile,9);
ThDD=ooMeshDD(meshpartfile);
u=@(x,y,z) cos(2*x-y-z).*sin(x-2*y+z);
U=ThDD.eval(u,'split',true);

Xi=ThDD.trace(U,'interface',true);

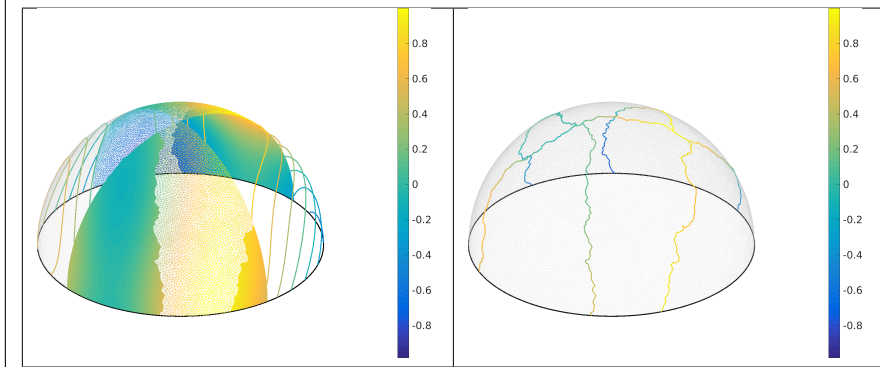
figure(1)
Th.plot('d',1,'color',[0,0,0])
Th.plot('d',2,'labels',[10,20,21], 'FaceColor','none', ...
    'EdgeColor',0.8*[1,1,1], 'EdgeAlpha',0.3)
ThDD.plotInterfacesVal(Xi,'FaceColor','interp', ...
    'EdgeColor','none','FaceAlpha',0.5)
ThDD.plotInterfacesIsolines(Xi,'format','%2f')
ThDD.plotInterfaces('bounds',true, 'color','k', ...
    'LineWidth',1)
axis image;axis off

figure(2)
Th.plot('d',1,'Color',[0,0,0])
idx=[1 2;2 3;9,7;6 1;6,4;8,2];
ThDD.plotPhysicals('d',2,'labels',[29,30,42,44,46,60,62,64], ...
    'FaceColor','none', 'EdgeColor',0.8*[1,1,1], ...
    'EdgeAlpha',0.3)
ThDD.plotInterfacesVal(Xi,'idx',idx,'FaceColor','interp', ...
    'EdgeColor','none')
ThDD.plotInterfacesIsolines(Xi,'idx',idx,'color',[0,0,0])
ThDD.plotInterfaces('idx',idx,'bounds',true, ...
    'color','k','LineWidth',1)
axis image;axis off

```

Listing 20: Data representations on interfaces of a 3D partitioned mesh

2.3.4 3D partitioned surface mesh



```

meshfile=gmsl.buildmesh3ds('demisphere4surf',50,'force',false);
meshpartfile=gmsl.buildpartmesh3ds(meshfile,6,'force',false);
Th=ooMeshDD(meshpartfile);

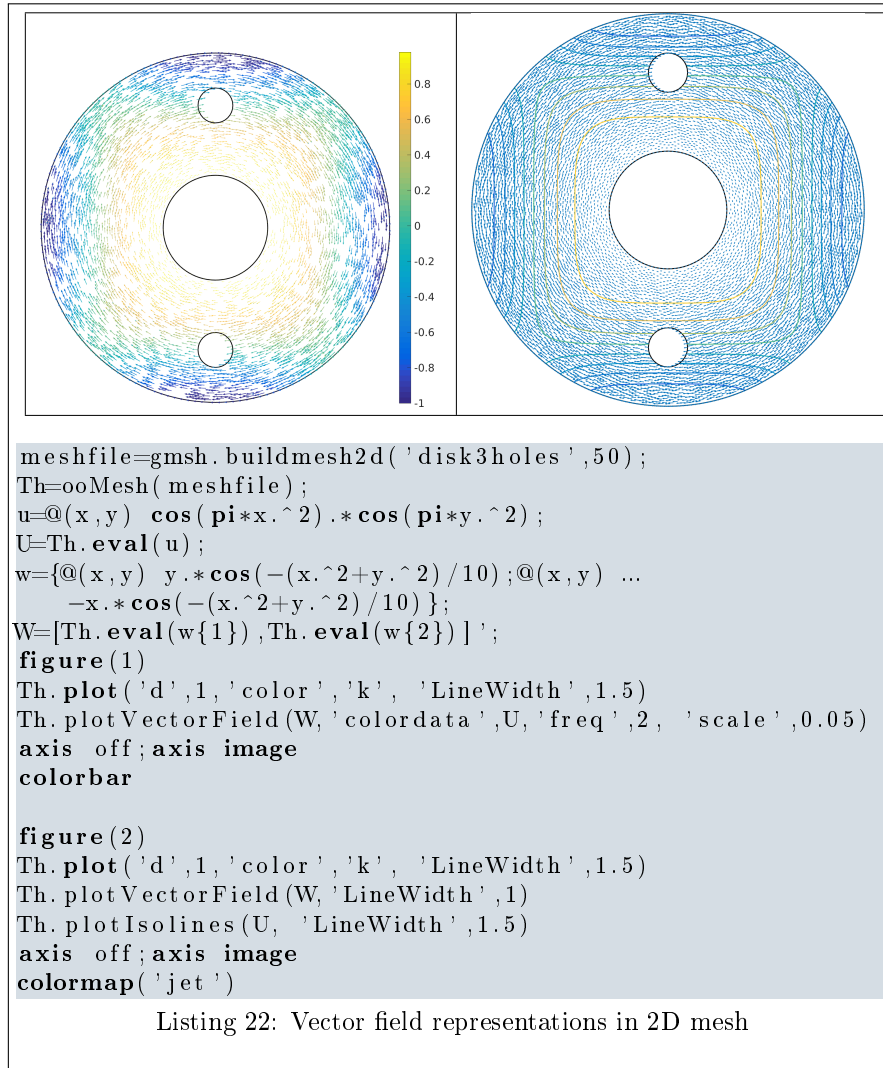
u=@(x,y,z) cos(2*x-y-z).*sin(x-2*y+z);
U=Th.eval(u);
isorange=linspace(min(U),max(U),10);
figure(1);
Th.plotVal(U,'FaceColor','interp','EdgeColor','none',...
    'labels',[1,3],'parts',1:2:6);
Th.plotIsolines(U,'labels',[1,3],'parts',1:2:6,...
    'isorange',isorange,'color','w');
Th.plotVal(U,'FaceColor','none','EdgeColor','interp',...
    'labels',[1,3],'parts',2:2:6);
Th.plotIsolines(U,'labels',[2,4],'isorange',isorange,...
    'LineWidth',2);
Th.plotPhysicals('labels',4,'FaceColor','None',...
    'EdgeColor',0.8*[1,1,1],'EdgeAlpha',0.3);
Th.plotPhysicals('d',1,'Color','k','LineWidth',2);
axis off;axis image
caxis([min(U),max(U)]);
colorbar

Xi=Th.trace(U,'interface',true);
figure(2)
Th.plotPhysicals('FaceColor','None',...
    'EdgeColor',0.8*[1,1,1],'EdgeAlpha',0.3);
Th.plotPhysicals('d',1,'color','k','LineWidth',2);
Th.plotInterfacesVal(Xi,'Linewidth',2);
axis off;axis image
caxis([min(U),max(U)]);
colorbar

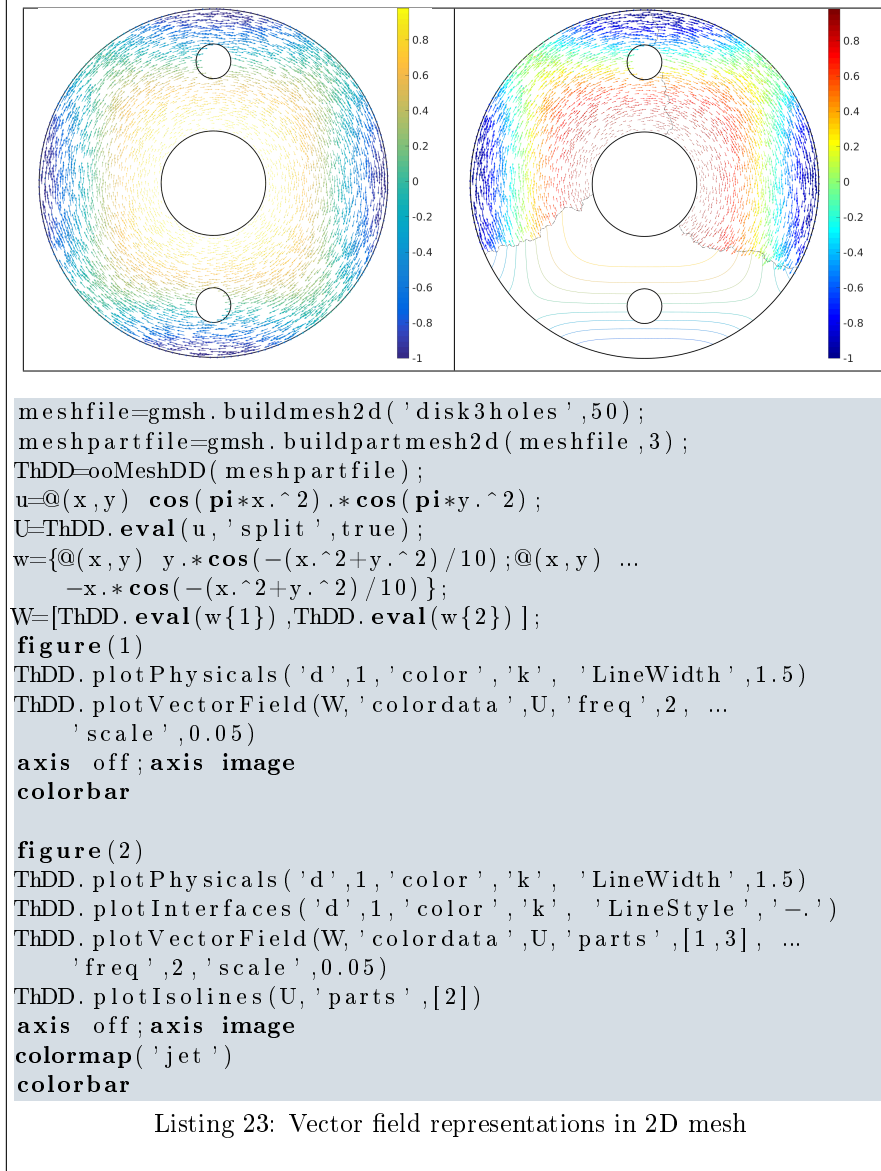
```

Listing 21: Data representations on partitioned surface mesh (left) and on its partitions interfaces

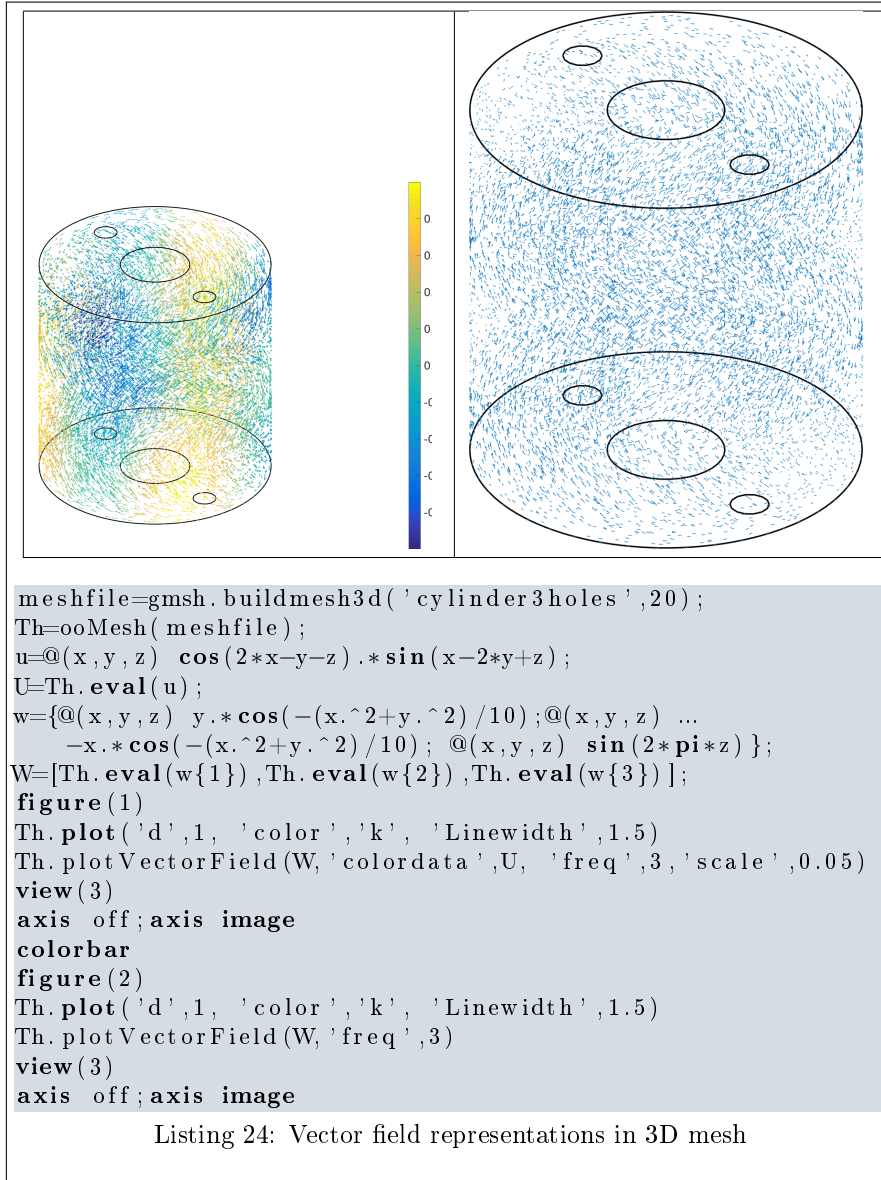
2.4.1 2D mesh



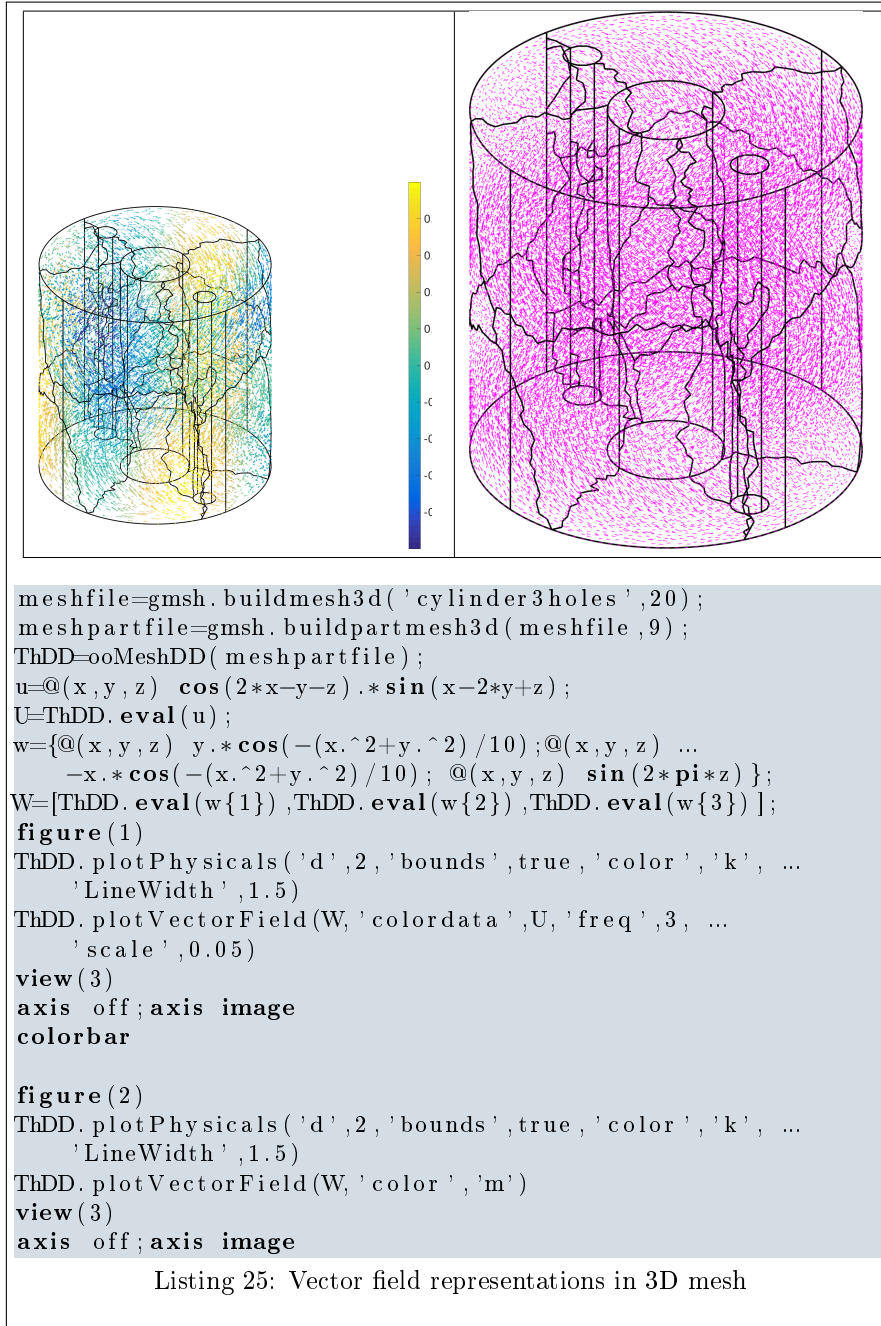
2.4.2 2D partitioned mesh



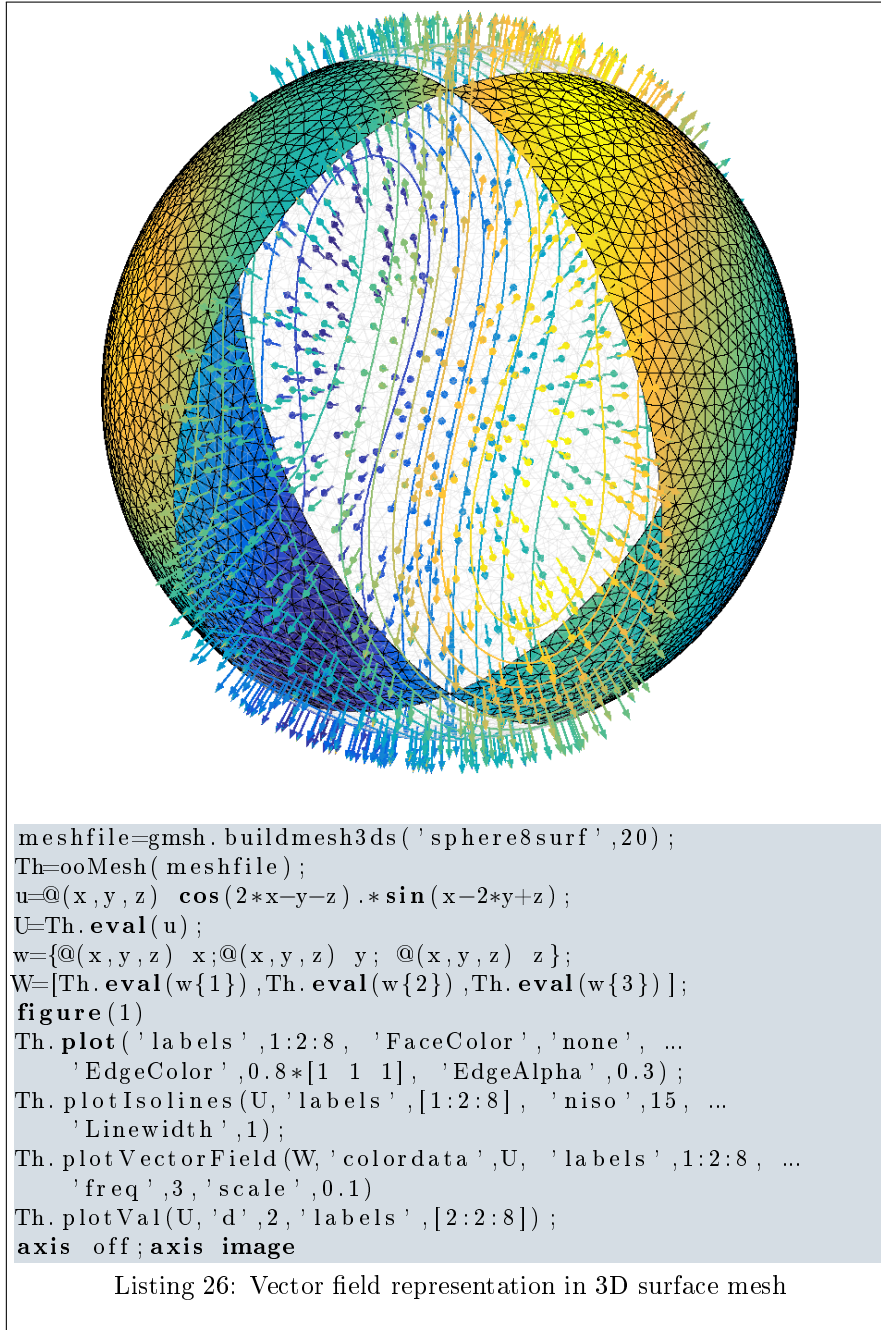
2.4.3 3D mesh



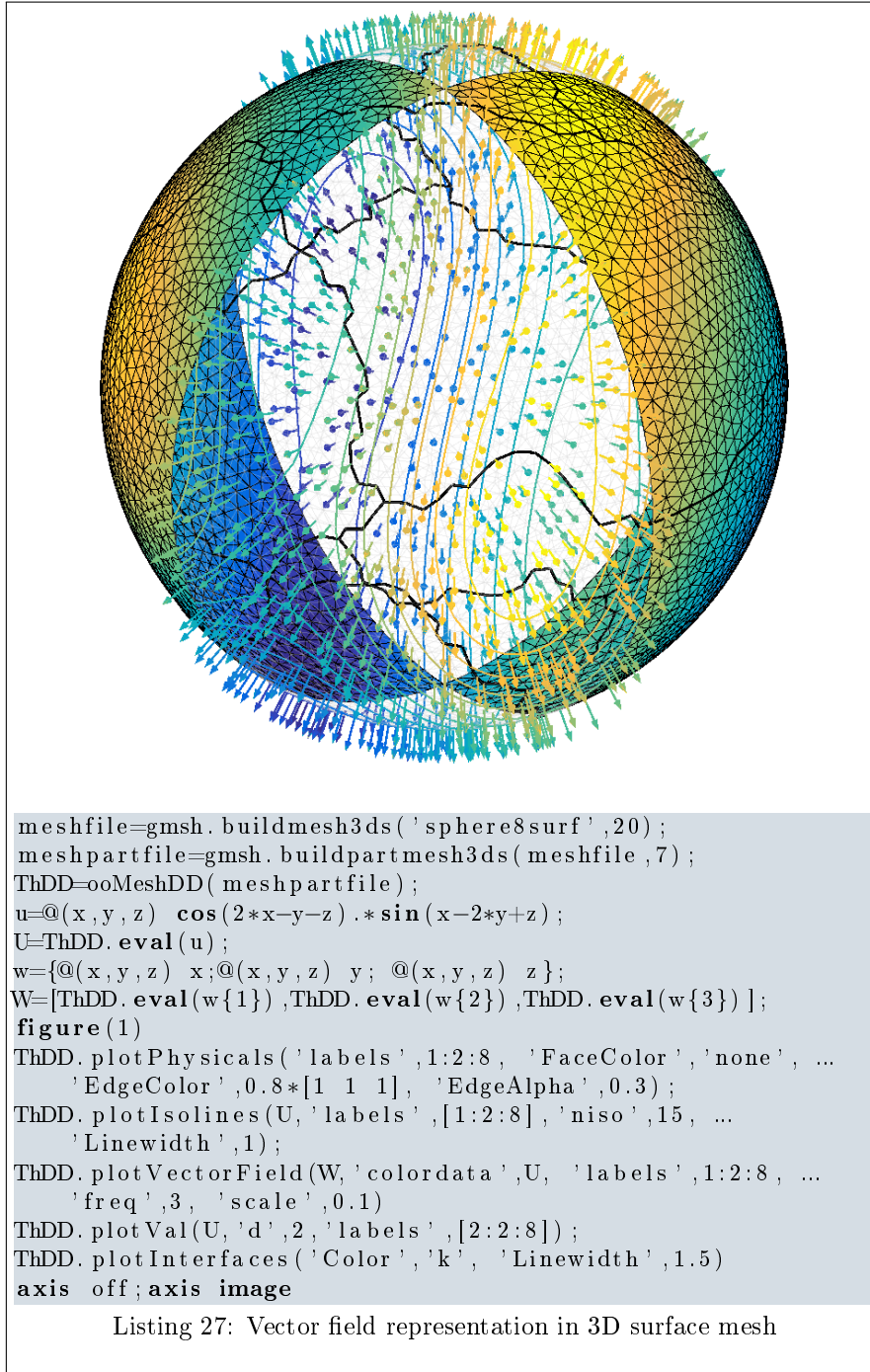
2.4.4 3D partitioned mesh



2.4.5 3D surface mesh



2.4.6 3D partitioned surface mesh



3.1 gmsh interface

gmsh is a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. It can also build two-dimensional meshes and three-dimensional surface meshes.

The source code and various pre-compiled versions of Gmsh (for Windows, Mac and Unix) can be downloaded from <http://gmsh.info>.

All the functions provided in this section use **gmsh** to create a mesh file from a **gmsh** geometry script file (extension *.geo*) which the object. . To read the mesh file, one can use **ooMesh** constructor or **ooMeshDD** constructor for a partitioned mesh. Some **gmsh** geometry script samples are given in directory **geodir** of the **mooMesh** toolbox.

In a first step we quickly present the installation of **gmsh** on Linux (for Windows and Mac OS X precompiled software are also provided on <http://gmsh.info>). Then, we explain how to configure the **mooMesh** toolbox for using **gmsh**. Finally, we describe the **mooMesh**'s functions which use **gmsh** to create mesh files.

3.1.1 gmsh installation (Linux 64bit)

```
mkdir -p ~/software/GMSH
cd ~/software/GMSH
wget http://gmsh.info/bin/Linux/gmsh-svn-Linux64.tgz
tar zxvf gmsh-svn-Linux64.tgz
```

To run **gmsh** we can use the following command in a terminal

```
~/software/GMSH/gmsh-2.12.1-svn-Linux/bin/gmsh &
```

3.1.2 Configuration in mooMesh toolbox

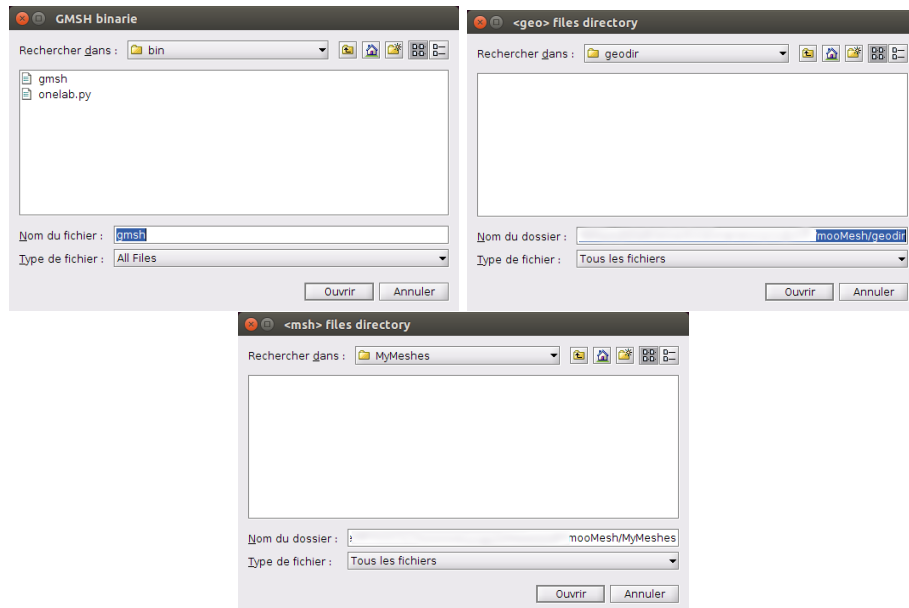
We must configure the toolbox for access to

- **gmsh** binary,
- **geo** files directory,
- **msh** files directory.

For that, under Matlab and in **mooMesh** toolbox directory, we do :

Command Window

```
>> InitmooMesh
>> mooMesh.configure
Configuration of mooMesh toolbox
-> Computer : hercule
-> User : cuvelier
1) Select gmsh binary ...
2) Select <geo> files directory ...
3) Select <msh> files directory ...
Write in ...
    /home/cuvelier/Travail/Recherche/CJS/VISUMESH/mooMesh/environment_hercule_cuvelier.m ...
-> done
fx >>
```



3.1.3 function `gmsh.buildmesh2d`

This function uses `gmsh` and a `.geo` file (describing a 2D-geometry) to generate a 2D-mesh.

Syntaxe

```
meshfile=gmsh.buildmesh2d(geofile,N)
meshfile=gmsh.buildmesh2d(geofile,N,Name,Value)
```

Description

`meshfile=gmsh.buildmesh2d(geofile,N)` create a 2D-mesh using `gmsh` and the `geo` file `geofile` (without path). The integer `N` has two functions : numbering the name of the generated mesh as `<geofile without extension and path> + <-N.msh>` and passing this number to `gmsh` via the option `"-setnumber N <N>".` Usually we used this parameter in `gmsh` to set the prescribed mesh element size at the points. (see given `geo` files)
As output return a file name (with full path) corresponding to the mesh generated by `gmsh`.

`meshfile=gmsh.buildmesh2d(geofile,N,Name,Value, ...)` specifies function options using one or more `Name,Value` pair arguments. The `Name` options can be

- `'geodir'` : to specify the directory of the `geo` file `geofile`,
- `'meshdir'` : to specify the directory where the mesh file will be written,

- **'meshfile'** : to specify the name of the mesh file (with path and .msh extension),
- **'check'** : to perform various consistency checks on mesh with gmsh, if Value is true. (default : false)
- **'force'** : to force meshing even if the mesh file already exists if Value is true (default : false)
- **'verbose'** : to specify the degree of verbosity (0, silence; 2, default; ...)
- **'strings'** : cells array of strings corresponding to gmsh options given with -string "... " (default empty) (see gmsh documentation)

Examples All the following examples use the .geo file **condenser11.geo** which is in the directory **geodir** of the toolbox.

Matlab commands with output

```
disp('****_gmsh.buildmesh2d_:1st_call')
meshfile=gmsh.buildmesh2d('condenser11',25,'force',true);
disp('****_gmsh.buildmesh2d_:2nd_call')
meshfile=gmsh.buildmesh2d('condenser11',25);

**** gmsh.buildmesh2d : 1st call
[mooMesh] Overwritting mesh file /home/cuvellier/Travail/Recherch/CJS/VISUMESH/mooMesh/MyMeshes/condenser11-25.msh
[mooMesh] Starting building mesh /home/cuvellier/Travail/Recherch/CJS/VISUMESH/mooMesh/MyMeshes/condenser11-25.msh with gmsh
2.12.1-svn
[mooMesh] Using command : /home/cuvellier/software/GMSH/gmsh-2.12.1-svn-Linux/bin/gmsh -2 -setnumber N 25 ...
/home/cuvellier/Travail/Recherch/CJS/VISUMESH/mooMesh/geodir/condenser11.geo -o ...
/home/cuvellier/Travail/Recherch/CJS/VISUMESH/mooMesh/MyMeshes/condenser11-25.msh
Be patient...
**** gmsh.buildmesh2d : 2nd call
[mooMesh] Mesh file /home/cuvellier/Travail/Recherch/CJS/VISUMESH/mooMesh/MyMeshes/condenser11-25.msh already exist.
-> Use "force" flag to rebuild if needed.
```

Matlab commands with output

```
meshfile=gmsh.buildmesh2d('condenser11',25,'force',true,...
    'verbose',4,'strings',{ 'Mesh.Algorithm=1; ', ...
    'Mesh.ScalingFactor=2; '});
```

```
[mooMesh] Overwritting mesh file /home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/MyMeshes/condenser11-25.msh
[mooMesh] Starting building mesh /home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/MyMeshes/condenser11-25.msh with gmesh
2.12.1-svn
[mooMesh] Using command : /home/cuvelier/software/GMSH/gmsh-2.12.1-svn-Linux/bin//gmsh -2 -setnumber N 25 -string ...
    "Mesh.Algorithm=1;Mesh.ScalingFactor=2;" ...
    /home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/geodir/condenser11.geo -o ...
    /home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/MyMeshes/condenser11-25.msh
Be patient...
[mooMesh] gmsh output :
Info : Running '/home/cuvelier/software/GMSH/gmsh-2.12.1-svn-Linux/bin//gmsh -2 -setnumber N 25 -string ...
    Mesh.Algorithm=1;Mesh.ScalingFactor=2; ...
    /home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/geodir/condenser11.geo -o ...
    /home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/MyMeshes/condenser11-25.msh' [gmsh 2.12.1-svn, 1 node, ...
    max. 1 thread]
Info : Started on Wed May 11 10:11:03 2016
Info : Reading '/home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/geodir/condenser11.geo'...
Info : Reading '/home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/geodir/options01_data.geo'...
Info : Done reading '/home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/geodir/options01_data.geo'
Info : Reading '/home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/geodir/shape_functions.geo'...
Info : Done reading '/home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/geodir/shape_functions.geo'
Info : Removing duplicate mesh vertices...
Info : Found 0 duplicate vertices
Info : No duplicate vertices found
Info : Done reading '/home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/geodir/condenser11.geo'
Info : Meshing 1D...
Info : Meshing curve 101 (Line)
Info : Meshing curve 102 (Line)
Info : Meshing curve 103 (Line)
Info : Meshing curve 104 (Line)
Info : Meshing curve 106 (Circle)
Info : Meshing curve 107 (Circle)
Info : Meshing curve 108 (Circle)
Info : Meshing curve 109 (Circle)
Info : Meshing curve 111 (Circle)
Info : Meshing curve 112 (Circle)
Info : Meshing curve 113 (Circle)
Info : Meshing curve 114 (Circle)
Info : Meshing curve 116 (Circle)
Info : Meshing curve 117 (Circle)
Info : Meshing curve 118 (Circle)
Info : Meshing curve 119 (Circle)
Info : Meshing curve 121 (Circle)
Info : Meshing curve 122 (Circle)
Info : Meshing curve 123 (Circle)
Info : Meshing curve 124 (Circle)
Info : Meshing curve 126 (Circle)
Info : Meshing curve 127 (Circle)
Info : Meshing curve 128 (Circle)
Info : Meshing curve 129 (Circle)
Info : Meshing curve 131 (Circle)
Info : Meshing curve 132 (Circle)
Info : Meshing curve 133 (Circle)
Info : Meshing curve 134 (Circle)
Info : Meshing curve 136 (Circle)
Info : Meshing curve 137 (Circle)
Info : Meshing curve 138 (Circle)
Info : Meshing curve 139 (Circle)
Info : Meshing curve 141 (Circle)
Info : Meshing curve 142 (Circle)
Info : Meshing curve 143 (Circle)
Info : Meshing curve 144 (Circle)
Info : Meshing curve 146 (Circle)
Info : Meshing curve 147 (Circle)
Info : Meshing curve 148 (Circle)
Info : Meshing curve 149 (Circle)
Info : Done meshing 1D (0.008537 s)
Info : Meshing 2D...
Info : Meshing surface 105 (Plane, MeshAdapt)
Info : Meshing surface 110 (Plane, MeshAdapt)
Info : Meshing surface 120 (Plane, MeshAdapt)
Info : Meshing surface 130 (Plane, MeshAdapt)
Info : Meshing surface 140 (Plane, MeshAdapt)
Info : Meshing surface 150 (Plane, MeshAdapt)
Info : Done meshing 2D (0.36489 s)
Info : 2999 vertices 6131 elements
Info : Writing '/home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/MyMeshes/condenser11-25.msh'...
Info : Done writing '/home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/MyMeshes/condenser11-25.msh'
Info : Stopped on Wed May 11 10:11:04 2016
```

3.1.4 function `gmsh.buildmesh3d`

This function uses `gmsh` and a `.geo` file (describing a 3D-geometry) to generate a 3D-mesh. See function `gmsh.buildmesh2d` for usage and options.

3.1.5 function `gmsh.buildmesh3ds`

This function uses `gmsh` and a `.geo` file (describing a 3D surface geometry or a 3D-geometry) to generate a 3D surface mesh. See function `gmsh.buildmesh2d` for usage and options.

3.1.6 function `gmsh.buildpartmesh2d`

This function uses `gmsh` and a `.msh` file (containing of a 2D-mesh) to generate a 2D partioned mesh.

Syntaxe

```
partmeshfile=gmsh.buildpartmesh2d(meshfile,np)
partmeshfile=gmsh.buildpartmesh2d(meshfile,np,Name,Value)
```

Description

`partmeshfile=gmsh.buildpartmesh2d(meshfile,np)` create a 2D partitioned mesh using `gmsh` and the `msh` file `meshfile` (with path). The integer `np` is the number of partitions.

As output return a file name (with full path) corresponding to the partitioned mesh generated by `gmsh`. The output file name is construct as following : `<meshfile without extension>-part<np>.msh`

`partmeshfile=gmsh.buildpartmesh2d(meshfile,np,Name,Value, ...)` specifies function options using one or more `Name,Value` pair arguments. The `Name` options can be

- `'savedir'` : to specify the directory where the partitioned mesh file will be written,
- `'check'` : to perform various consistency checks on mesh with `gmsh`, if `Value` is true. (default : false)
- `'force'` : to force meshing even if the mesh file already exists if `Value` is true (default : false)
- `'verbose'` : to specify the degree of verbosity (0, silence; 2, default; ...)
- `'strings'` : cells array of strings corresponding to `gmsh` options given with `-string "..."` (default empty) (see `gmsh` documentation)

Examples All the following examples use the `meshfile` as output of the command :

```
meshfile=gmsht.buildmesh2d('condenser11',25);
```

Matlab commands with output

```
pmfile=gmsht.buildpartmesh2d(meshfile,5,'force',true);
```

```
[mooMesh] Overwriting mesh file /home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/MyMeshes/condenser11-25-part5.msh ...
[mooMesh] Starting building mesh /home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/MyMeshes/condenser11-25-part5.msh ...
           with gmsht
2.12.1-svn
[mooMesh] Using command : /home/cuvelier/software/GMSH/gmsh-2.12.1-svn-Linux/bin//gmsh -2 -saveall -part 5 ...
           /home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/MyMeshes/condenser11-25.msh -o ...
           /home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/MyMeshes/condenser11-25-part5.msh
Be patient...
```

Matlab commands with output

```
pmfile=gmsht.buildpartmesh2d(meshfile,5,'force',true, ...
    'verbose',4,'strings',{'Mesh.Partitioner=2;',' ...
    'Mesh.MetisAlgorithm=3;'});
```

```
[mooMesh] Overwriting mesh file /home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/MyMeshes/condenser11-25-part5.msh ...
[mooMesh] Starting building mesh /home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/MyMeshes/condenser11-25-part5.msh ...
           with gmsht
2.12.1-svn
[mooMesh] Using command : /home/cuvelier/software/GMSH/gmsh-2.12.1-svn-Linux/bin//gmsh -2 -saveall -part 5 -string ...
           "Mesh.Partitioner=2;Mesh.MetisAlgorithm=3;" ...
           /home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/MyMeshes/condenser11-25.msh -o ...
           /home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/MyMeshes/condenser11-25-part5.msh
Be patient...
[mooMesh] gmsht output :
Info : Running '/home/cuvelier/software/GMSH/gmsh-2.12.1-svn-Linux/bin//gmsh -2 -saveall -part 5 -string ...
           Mesh.Partitioner=2;Mesh.MetisAlgorithm=3; ...
           /home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/MyMeshes/condenser11-25.msh -o ...
           /home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/MyMeshes/condenser11-25-part5.msh' [Gmsh 2.12.1-svn, 1 ...
           node, max. 1 thread]
Info : Started on Wed May 11 11:27:45 2016
Info : Reading '/home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/MyMeshes/condenser11-25.msh'...
Info : 2990 vertices
Info : Vertex numbering is dense
Info : 6082 elements
Info : Done reading '/home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/MyMeshes/condenser11-25.msh'
Info : Meshing 1D...
Info : Done meshing 1D (2.9e-05 s)
Info : Meshing 2D...
Info : Done meshing 2D (2.7895e-05 s)
Info : 2990 vertices 6082 elements
Info : Building graph...
Info : Partitioning graph...
Info : Launching METIS graph partitioner
METIS with weights
Info : Number of Edges Cut : 133
Info : Done partitioning graph
Info : Writing '/home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/MyMeshes/condenser11-25-part5.msh'...
Info : Done writing '/home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/MyMeshes/condenser11-25-part5.msh'
Info : Stopped on Wed May 11 11:27:45 2016
```

3.1.7 function gmsht.buildpartmesh3d

This function uses `gmsht` and a `.msh` file (containing of a 3D-mesh) to generate a 3D partioned mesh.

3.1.8 function gmsht.buildpartmesh3ds

This function uses `gmsht` and a `.msh` file (containing of a 3D surface mesh) to generate a 3D partioned surface mesh.

3.1.9 function gmsh.buildPartRectangle

This function uses `gmsh` and the `geodir/rectanglepart.geo` file to generate a 2D regular partitioned mesh of the rectangle $[0, Lx] \times [0, Ly]$ with $Nx \times Ny$ partitions.

Syntaxe

```
meshfile=gmsh.buildpartrectangle(Lx,Ly,Nx,Ny,N)

meshfile=gmsh.buildpartrectangle(Lx,Ly,Nx,Ny,N, ...
    Name,Value)
```

Description

`meshfile=gmsh.buildpartrectangle(Lx,Ly,Nx,Ny,N)` create a 2D regular partitioned mesh using `gmsh` of the rectangle $[0, Lx] \times [0, Ly]$ with $Nx \times Ny$ partitions. The `N` parameter is passed to `gmsh` to set the prescribed mesh element size at the points

As output return a file name (with full path) corresponding to the partitioned mesh generated by `gmsh`. The default output file name is construct as following : `rectanglepart-Lx%.3f-Ly%.3f-Nx%d-Ny%d-N%d.msh`

`meshfile=gmsh.buildpartrectangle(Lx,Ly,Nx,Ny,N,Name,Value, ...)` specifies function options using one or more `Name,Value` pair arguments. The `Name` options can be

- `'meshdir'` : to specify the directory where the partitioned mesh file will be written,
- `'meshfile'` : to specify the mesh file name with `.msh` extension. Without path, the file is written in `<meshdir>` directory.
- `'check'` : to perform various consistency checks on mesh with `gmsh`, if `Value` is true. (default : false)
- `'force'` : to force meshing even if the mesh file already exists if `Value` is true (default : false)
- `'verbose'` : to specify the degree of verbosity (0, silence; 2, default; ...)
- `'strings'` : cells array of strings corresponding to `gmsh` options given with `-string "..."` (default empty) (see `gmsh` documentation)

Examples All the following examples ...

Matlab commands with output

```
pmfile=gmshtool.buildpartrectangle(1,1,3,2,100,'force',true);
```

```
[mooMesh] Overwriting mesh file ...
/home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/MyMeshes/rectanglepart-Lx1.000-Ly1.000-Nx3-Ny2-M100.msh
[mooMesh] Starting building mesh ...
/home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/MyMeshes/rectanglepart-Lx1.000-Ly1.000-Nx3-Ny2-M100.msh ...
with gmshtool
2.12.1-svn
[mooMesh] Using command : /home/cuvelier/software/GMSH/gmsh-2.12.1-svn-Linux/bin/gmsh -2 -setnumber N 100 -setnumber ...
NX 3 -setnumber NY 2 -setnumber LX 1 -setnumber LY 1 ...
/home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/geodir/rectanglepart.geo -o ...
/home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/MyMeshes/rectanglepart-Lx1.000-Ly1.000-Nx3-Ny2-M100.msh
Be patient...
```

Matlab commands with output

```
pmfile=gmshtool.buildpartrectangle(1,1,3,2,100,'verbose',4, ...
'force',true,'meshfile','./toto.msh');
```

```
[mooMesh] Overwriting mesh file ./toto.msh
[mooMesh] Starting building mesh ./toto.msh with gmshtool
2.12.1-svn
[mooMesh] Using command : /home/cuvelier/software/GMSH/gmsh-2.12.1-svn-Linux/bin/gmsh -2 -setnumber N 100 -setnumber ...
NX 3 -setnumber NY 2 -setnumber LX 1 -setnumber LY 1 ...
/home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/geodir/rectanglepart.geo -o ./toto.msh
Be patient...
[mooMesh] gmshtool output :
Info : Running '/home/cuvelier/software/GMSH/gmsh-2.12.1-svn-Linux/bin/gmsh -2 -setnumber N 100 -setnumber NX 3 ...
-setnumber NY 2 -setnumber LX 1 -setnumber LY 1 ...
/home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/geodir/rectanglepart.geo -o ./toto.msh' [Gmsh 2.12.1-svn, ...
1 node, max. 1 thread]
Info : Started on Thu May 12 07:29:00 2016
Info : Reading '/home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/geodir/rectanglepart.geo'...
Info : Reading '/home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/geodir/partitions01.data.geo'...
Info : Done reading '/home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/geodir/partitions01.data.geo'
Info : Reading '/home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/geodir/partitions_shape.geo'...
Info : Done reading '/home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/geodir/partitions_shape.geo'
Info : Done reading '/home/cuvelier/Travail/Recherch/CJS/VISUMESH/mooMesh/geodir/rectanglepart.geo'
Info : Meshing 1D...
Info : Meshing curve 1 (Line)
Info : Meshing curve 2 (Line)
Info : Meshing curve 3 (Line)
Info : Meshing curve 4 (Line)
Info : Meshing curve 5 (Line)
Info : Meshing curve 6 (Line)
Info : Meshing curve 7 (Line)
Info : Meshing curve 8 (Line)
Info : Meshing curve 9 (Line)
Info : Meshing curve 10 (Line)
Info : Meshing curve 11 (Line)
Info : Meshing curve 12 (Line)
Info : Meshing curve 13 (Line)
Info : Meshing curve 14 (Line)
Info : Meshing curve 15 (Line)
Info : Meshing curve 16 (Line)
Info : Meshing curve 17 (Line)
Info : Done meshing 1D (0.00148 s)
Info : Meshing 2D...
Info : Meshing surface 19 (Plane, Delaunay)
Info : Meshing surface 21 (Plane, Delaunay)
Info : Meshing surface 23 (Plane, Delaunay)
Info : Meshing surface 25 (Plane, Delaunay)
Info : Meshing surface 27 (Plane, Delaunay)
Info : Meshing surface 29 (Plane, Delaunay)
Info : Done meshing 2D (0.569366 s)
Info : 13685 vertices 27682 elements
Info : Writing './toto.msh'...
Info : Done writing './toto.msh'
Info : Stopped on Thu May 12 07:29:01 2016
```

3.2 ooMesh methods

3.2.1 ooMesh constructor

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

or ... (triangle).

Syntaxe

```
Th=ooMesh(meshfile)

Th=ooMesh(meshfile,Name,Value)
```

Description

`Th=ooMesh(meshfile)` create the `mooMesh` object `Th` from the mesh file `meshfile` (`gmsh` format by default).

Th=ooMesh(meshfile,Name,Value, ...) specifies function options using one or more Name,Value pair arguments. The Name options can be

- **'format'**: to specify the format of the mesh file `meshfile`. Value must be `'medit'`, `'gmsht'` (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 `.geo` file `condenser11.geo` which is in the directory `geodir` of the toolbox.

```
Matlab commands with output
```

```
meshfile=gmshtool.buildmesh2d('condenser11',25,'verbose',0);  
disp('***_Read_mesh_***')  
Th=ooMesh(meshfile)  
  
*** Read mesh ***  
  
Th =  
  
ooMesh with properties:  
  
    d: 2  
   dim: 2  
    sTh: {1x19 cell}  
   nsTh: 19  
toGlobal: [1x3474 double]  
toParent: [1x3474 double]  
   sThimp: [1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2]  
   sThlab: [1 2 3 4 5 6 7 8 20 101 102 103 104 2 4 6 8 10 20]  
   sThcolor: [19x3 double]  
      bbox: [-1 1 -1 1]  
   sThgeolab: []  
   sThphyslab: [2 4 6 8 10 20]  
   sThpartlabs: {}  
           M: [7050x55 int32]  
          Ms: [1x2 struct]  
         na: 3474
```

3.2.2 function plot

The method `plot` displays the mesh or parts of the mesh defined by an `ooMesh` object.

Syntaxe

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

Description

`Th.plot()` displays all the `Th.d`-dimensional simplices elements.

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

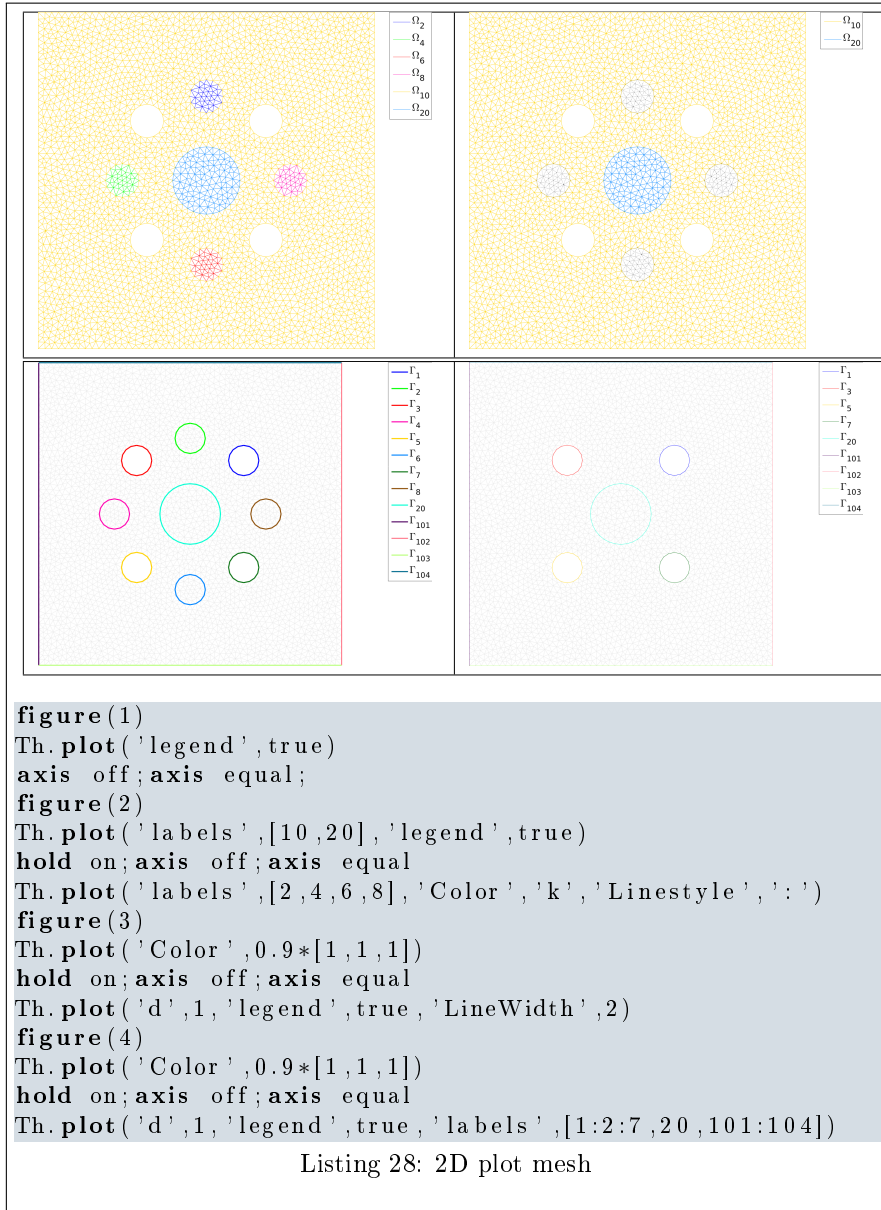
- `'d'` : to specify the dimension of the simplices elements (default : `Th.d`)
- `'labels'` : to select the labels of the elements to display,
- `'color'` : to specify the color of the displayed mesh elements. (default : use one color by displayed mesh elements),
- `'legend'` : add a legend to graph if true (default : false)
- `'bounds'` : If true, draw the borders of each one of the selected elementaries mesh elements (only for 2-dimensional simplices). (default : false)

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

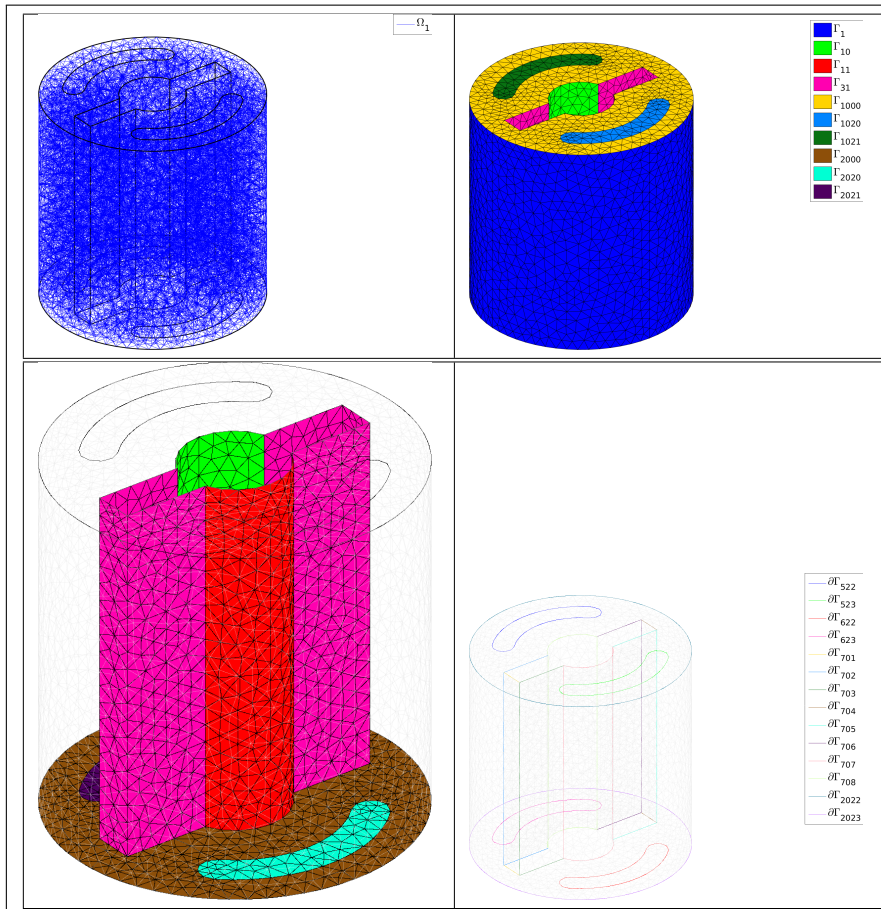
One can use any option of the following functions according to the type of d -simplex to be represented.

- In dimension 3,
 - if $d == 3$, `tetramesh` function is used,
 - if $d == 2$, `trimesh` function is used,
 - if $d == 1$, `line` function is used,
 - if $d == 0$, `plot3` function is used,
- In dimension 2,
 - if $d == 2$, `triplot` function is used,
 - if $d == 1$, `line` 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.



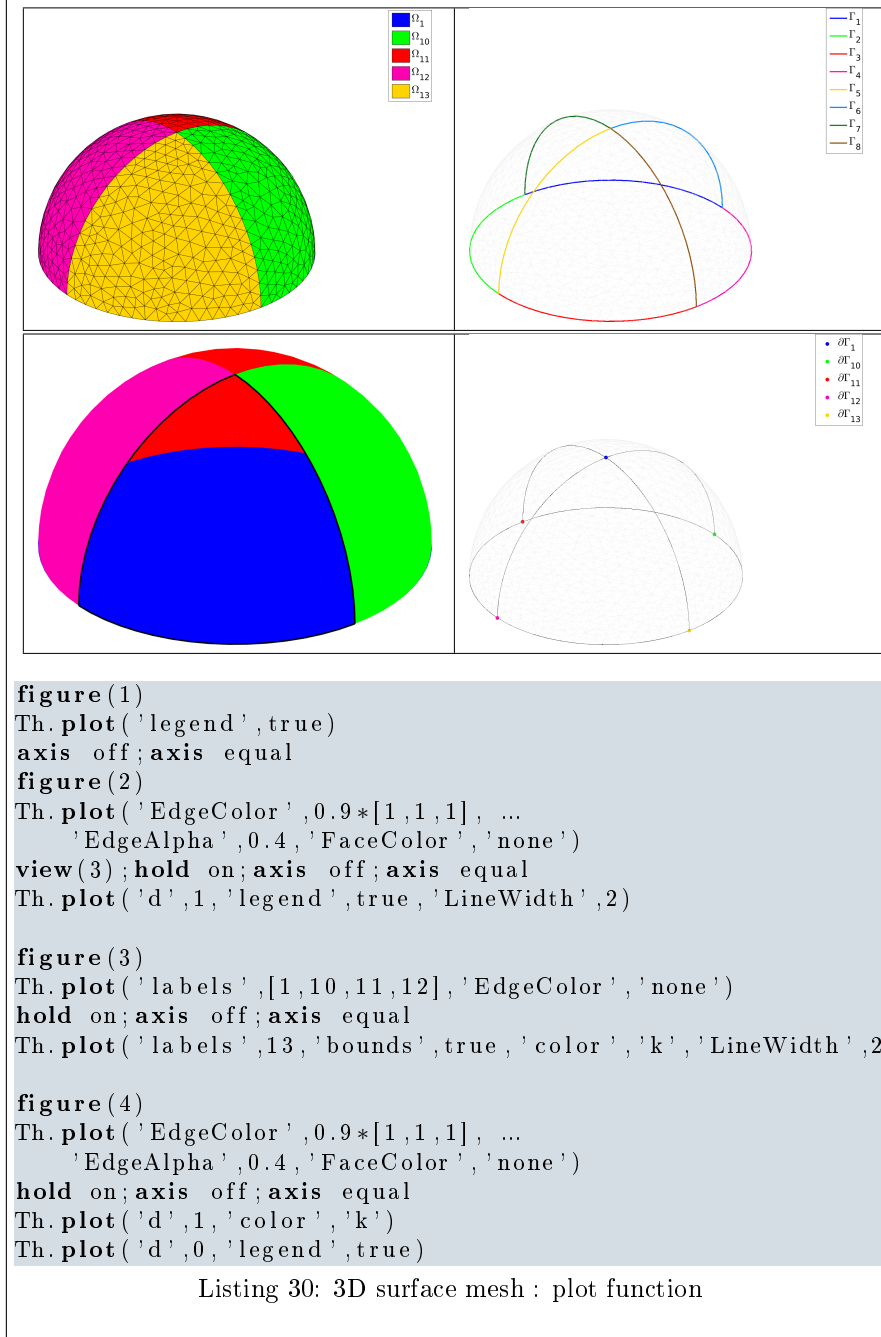
```

figure(1)
Th.plot('legend',true)
hold on
Th.plot('d',1,'Color','k','Linewidth',1.5)
axis off;axis equal
figure(2)
Th.plot('d',2,'legend',true)
view(3);hold on;axis off;axis equal
figure(3)
Th.plot('d',2,'labels',[1,1000,1020,1021], ...
        'EdgeColor',0.9*[1,1,1], ...
        'EdgeAlpha',0.4,'FaceColor','none')
hold on;axis off;axis equal
Th.plot('d',2,'labels',1000,'bounds',true,'color','k')
Th.plot('d',2,'labels',[10,11,31,2000,2020,2021])
figure(4)
Th.plot('d',2,'EdgeColor',0.9*[1,1,1], ...
        'EdgeAlpha',0.4,'FaceColor','none')
hold on;axis off;axis equal
Th.plot('d',1,'legend',true)

```

Listing 29: 3D plot mesh

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.



3.2.3 function plotVal

The method `plotVal` displays datas on the mesh or parts of the mesh defined by an `ooMesh` object.

Syntaxe

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

Description

`Th.plotVal(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`.

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

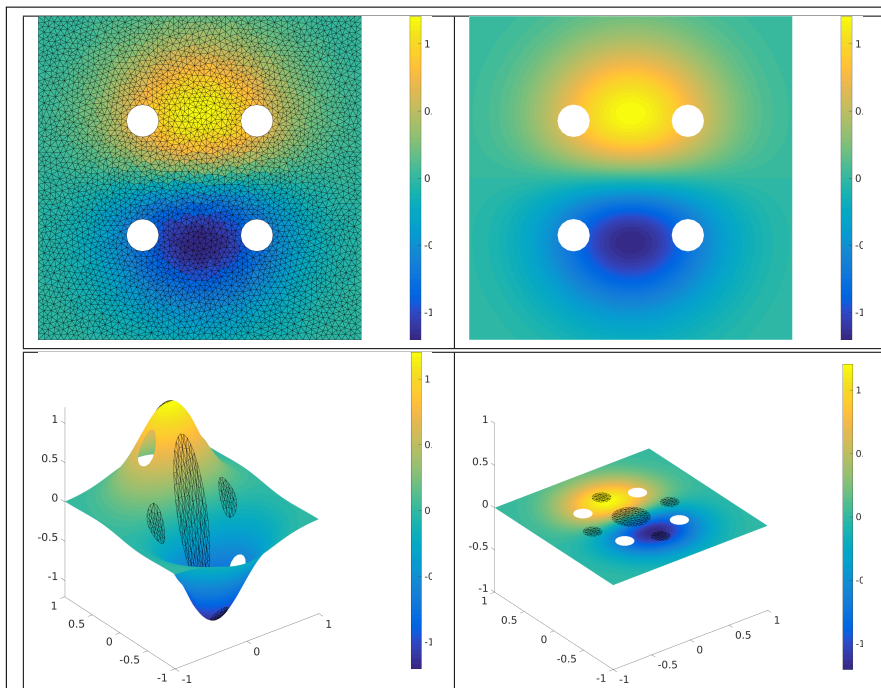
- `'d'` : to specify the dimension of the simplices elements (default : `Th.d`)
- `'labels'` : to select the labels of the elements to display data,
- `'plan'` : if true, (default : false)

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 `'plan'` is true, **patch** function is used, otherwise **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.



```

u=Th.eval(@(x,y) 5*exp(-3*(x.^2+y.^2)).*cos(x).*sin(y));
figure(1)
Th.plotVal(u)
axis off; axis equal;
colorbar
figure(2)
Th.plotVal(u)
axis off; axis equal;
shading interp; colorbar

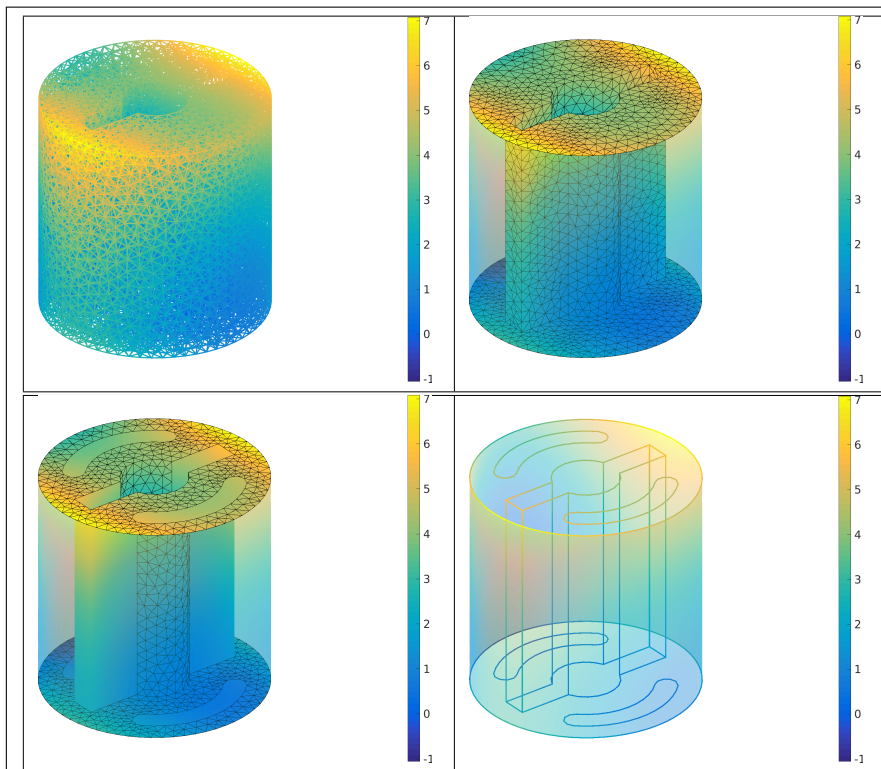
figure(3)
Th.plotVal(u,'labels',[2:2:8,20],'FaceColor','interp')
view(3); hold on;
Th.plotVal(u,'labels',10,'FaceColor','interp','EdgeColor','none')
axis equal; colorbar

figure(4)
Th.plotVal(u,'labels',[2:2:8,20],'plan',true ...
,'FaceColor','interp')
view(3); hold on;
Th.plotVal(u,'labels',10,'plan',true, ...
,'FaceColor','interp','EdgeColor','none')
axis equal; colorbar

```

Listing 31: 2D mesh : plotVal function

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.



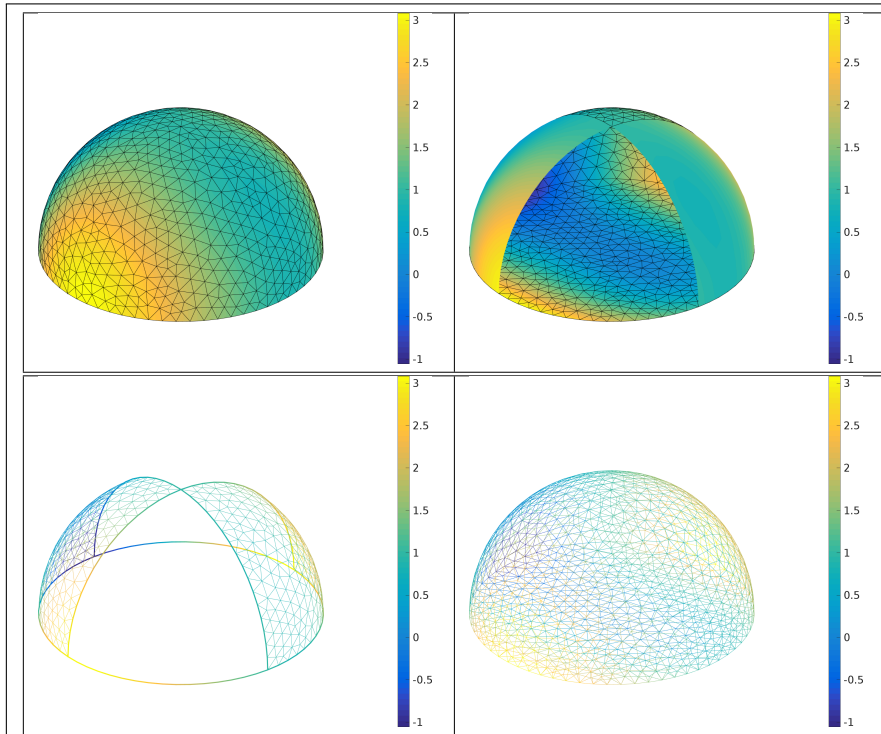
```

u=Th.eval(@(x,y,z) 3*x.^2-y.^2+3*z.^2+x.*y);
figure(1)
Th.plotVal(u)
axis off;axis equal;colorbar
figure(2)
Th.plotVal(u,'d',2,'labels',[10,11,31,1000,1020,1021,2000,2020,2021])
hold on
Th.plotVal(u,'d',2,'labels',1,'FaceColor','interp', ...
    'EdgeColor','none','FaceAlpha',0.4)
axis off;axis equal;colorbar
figure(3)
Th.plotVal(u,'d',2,'labels',1,'FaceColor','interp', ...
    'EdgeColor','none','FaceAlpha',0.4)
hold on
Th.plotVal(u,'d',2,'labels',[10,11,1000,2000])
Th.plotVal(u,'d',2,'labels',[31,1020,1021,2020,2021], ...
    'FaceColor','interp','EdgeColor','none')
axis off;axis equal;colorbar
figure(4)
Th.plotVal(u,'d',2,'labels',1,'FaceColor','interp', ...
    'EdgeColor','none','FaceAlpha',0.4)
hold on
Th.plotVal(u,'d',1,'LineWidth',2)
axis off;axis equal;colorbar

```

Listing 32: 3D mesh : plotVal function

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.



```
u=Th.eval(@(x,y,z) 3*x.^2-y.^3+z.^2+x.*y);
figure(1)
Th.plotVal(u)
axis off;axis equal;colorbar;
figure(2)
Th.plotVal(u,'labels',[1,11])
hold on
Th.plotVal(u,'labels',[10,12], 'FaceColor','interp', ...
'EdgeColor','none')
axis off;axis equal;colorbar;
figure(3)
Th.plotVal(u,'d',1,'LineWidth',2)
hold on;axis off;axis equal;colorbar;
Th.plotVal(u,'labels',[10,12], 'FaceColor','none', ...
'EdgeColor','interp')
figure(4)
Th.plotVal(u,'FaceColor','none', 'EdgeColor','interp')
axis off;axis equal;colorbar;
```

Listing 33: 3D surface mesh : `plotVal` function

3.2.4 function plotIsolines

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

Syntaxe

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

Description

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

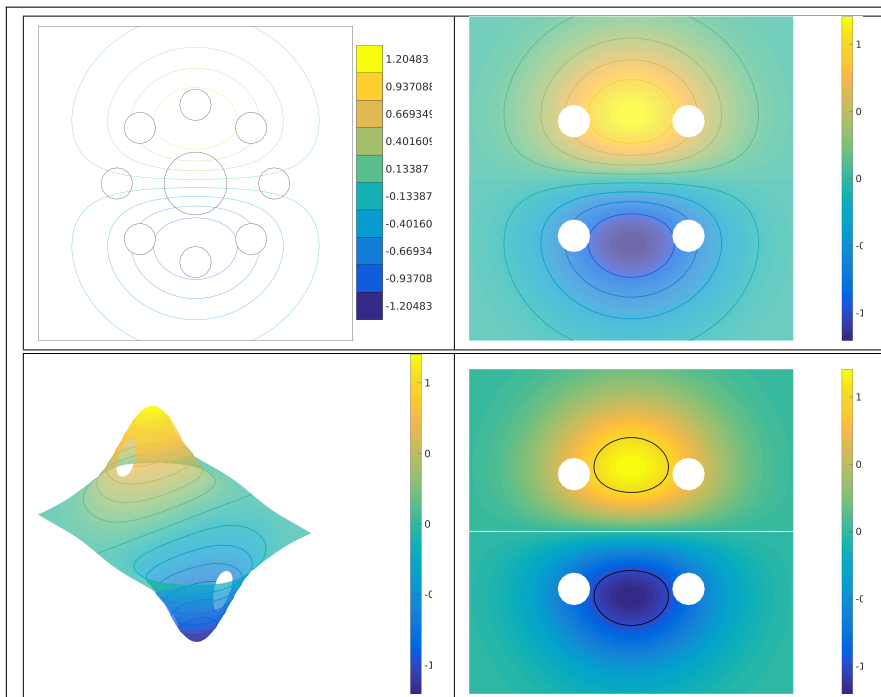
`Th.plotVal(u,Name,Value, ...)` specifies function options using one or more `Name,Value` pair arguments. Options of first level are

- `'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,
- `'plan'` : if true, (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 `'plan'` set to false
- `plot` function in 2 with `'plan'` set to true

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



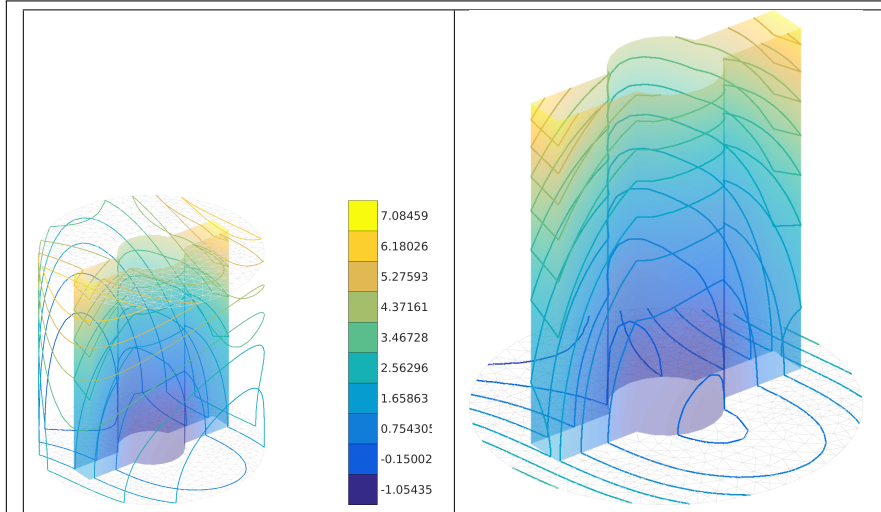
```

u=Th.eval(@(x,y) 5*exp(-3*(x.^2+y.^2)).*cos(x).*sin(y));
figure(1)
Th.plot('d',1,'color','k')
hold on
Th.plotIsolines(u,'isocolorbar',true)
axis off;axis equal;
figure(2)
Th.plotVal(u,'plan',true,'FaceAlpha',0.7)
shading interp;hold on
Th.plotIsolines(u,'plan',true,'LineWidth',1.5)
axis off;axis equal;
colorbar
figure(3)
Th.plotVal(u,'FaceAlpha',0.7)
view(3)
shading interp;hold on
Th.plotIsolines(u,'niso',15,'LineWidth',1.5)
axis off;axis equal;
colorbar
figure(4)
Th.plotVal(u,'plan',true)
shading interp;hold on
Th.plotIsolines(u,'isorange',0,'LineWidth',1.5,...
    'color','w')
Th.plotIsolines(u,'isorange',[-1,1], 'LineWidth',1.5,...
    'color','k','plan',true)
axis off;axis equal;colorbar

```

Listing 34: 2D mesh : plotIsolines function

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.



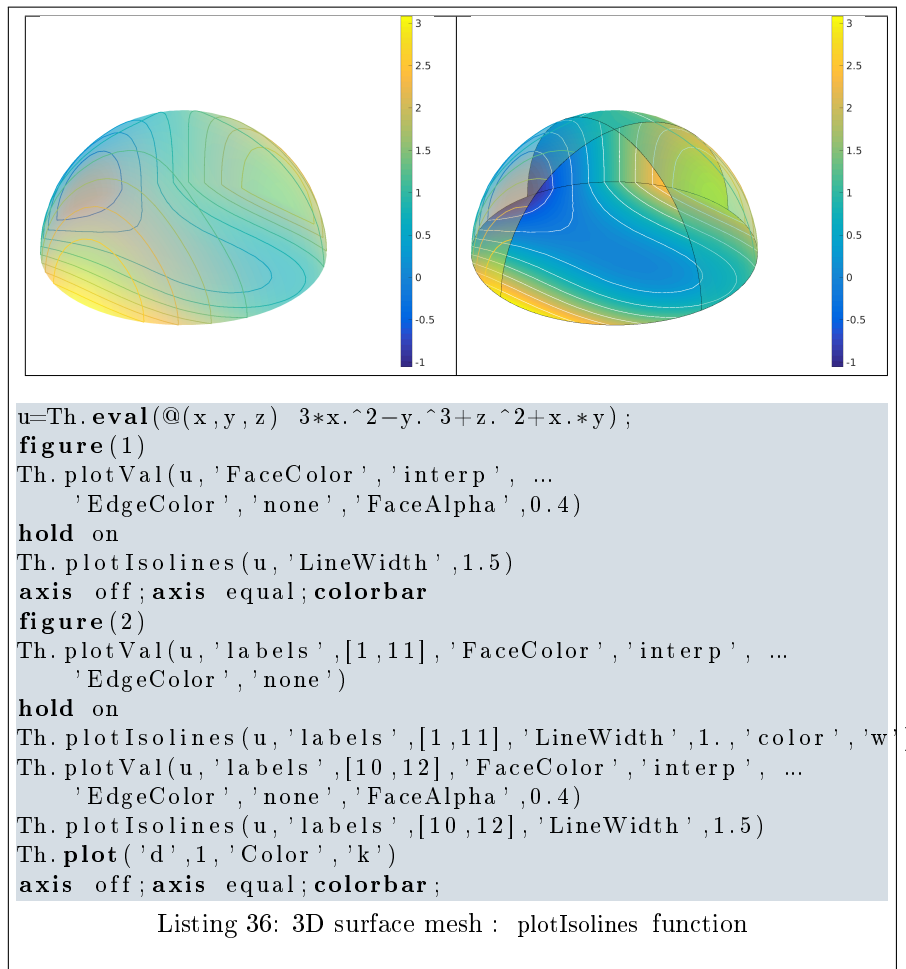
```

u=Th.eval(@(x,y,z) 3*x.^2-y.^3+z.^2+x.*y);
figure(1)
Th.plotVal(u,'d',2,'labels',[10,11,31],'FaceColor','interp', ...
'EdgeColor','none','FaceAlpha',0.4)
hold on
Th.plot('d',2,'labels',[1000,1020,1021,2000,2020,2021], ...
'FaceColor','none','EdgeColor',0.9*[1,1,1])
Th.plotIsolines(u,'isocolorbar',true,'LineWidth',1.5)
view(3);axis off;axis equal;
figure(2)
Th.plotVal(u,'d',2,'labels',[10,11,31],'FaceColor','interp', ...
'EdgeColor','none','FaceAlpha',0.4)
hold on
Th.plot('d',2,'labels',[2000,2020,2021], ...
'FaceColor','none','EdgeColor',0.9*[1,1,1])
Th.plotIsolines(u,'labels',[10,11,31,2000,2020,2021],'LineWidth',1.5, ...
'niso',15)
axis off;axis equal;

```

Listing 35: 3D mesh : `plotIsolines` function

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.



3.2.5 function slicemesh

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

Syntaxe

```

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

```

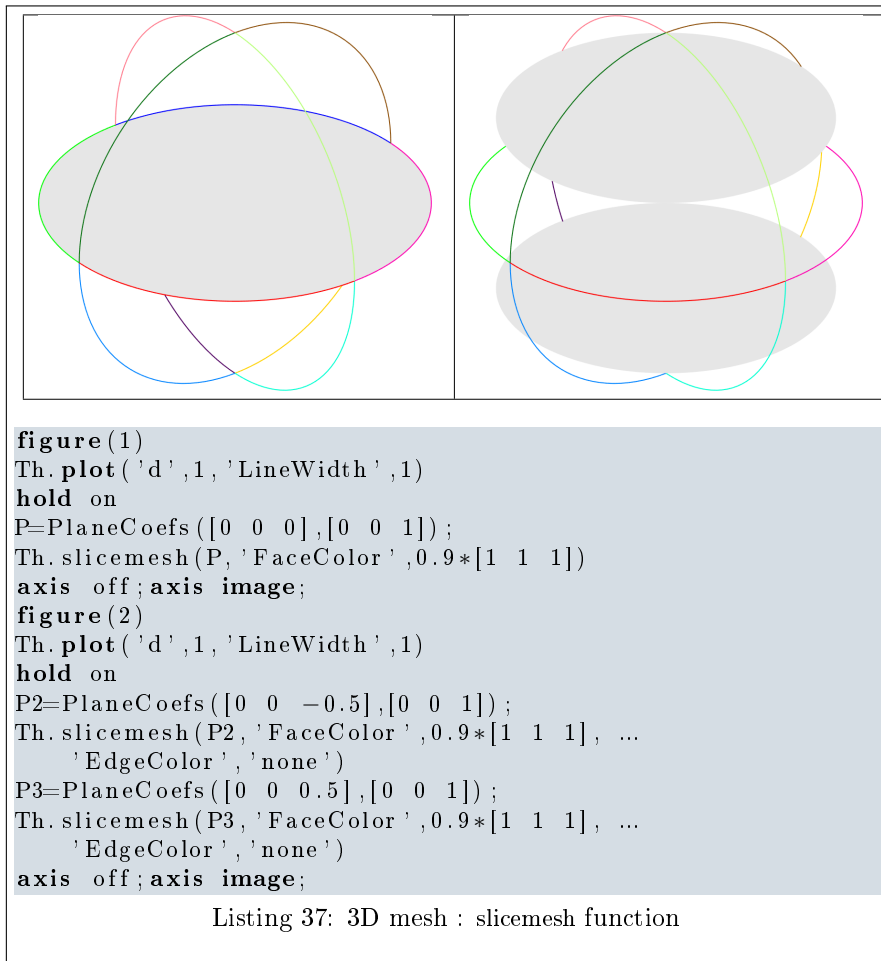
Description

`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. To compute P one can use the function `PlaneCoefs` of the `mooMesh` toolbox. With this function, the array P , is obtained with $P = \text{PlaneCoefs}(Q, V)$ where Q is a point in the plane and V is a vector orthogonal to it.

`Th.plotVal(u,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 : light grey, `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.



3.2.6 function slice

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

Syntaxe

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

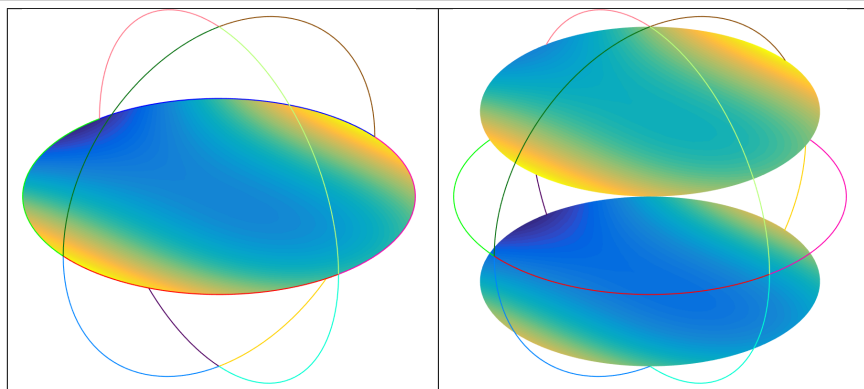
Description

`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. 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 `PlaneCoefs` of the `mooMesh` toolbox. With this function, the array `P`, is obtained with `P=PlaneCoefs(Q,V)` where `Q` is a point in the plane and `V` is a vector orthogonal to it.

`Th.slide(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.



```
u=Th.eval(@(x,y,z) 3*x.^2-y.^3+z.^2+x.*y+z);
figure(1)
Th.plot('d',1,'LineWidth',1)
hold on
P=PlaneCoefs([0 0 0],[0 0 1]);
Th.slice(u,P,'Facecolor','interp')
axis off;axis image;
figure(2)
Th.plot('d',1,'LineWidth',1)
hold on
P2=PlaneCoefs([0 0 -0.5],[0 0 1]);
Th.slice(u,P2,'Facecolor','interp')
P3=PlaneCoefs([0 0 0.5],[0 0 1]);
Th.slice(u,P3,'Facecolor','interp')
axis off;axis image;
```

Listing 38: 3D mesh : `slice` function

3.2.7 function isoslice

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

Syntaxe

```
Th.isoslice(u,P)
Th.isoslice(u,P,Name,Value, ...)
```

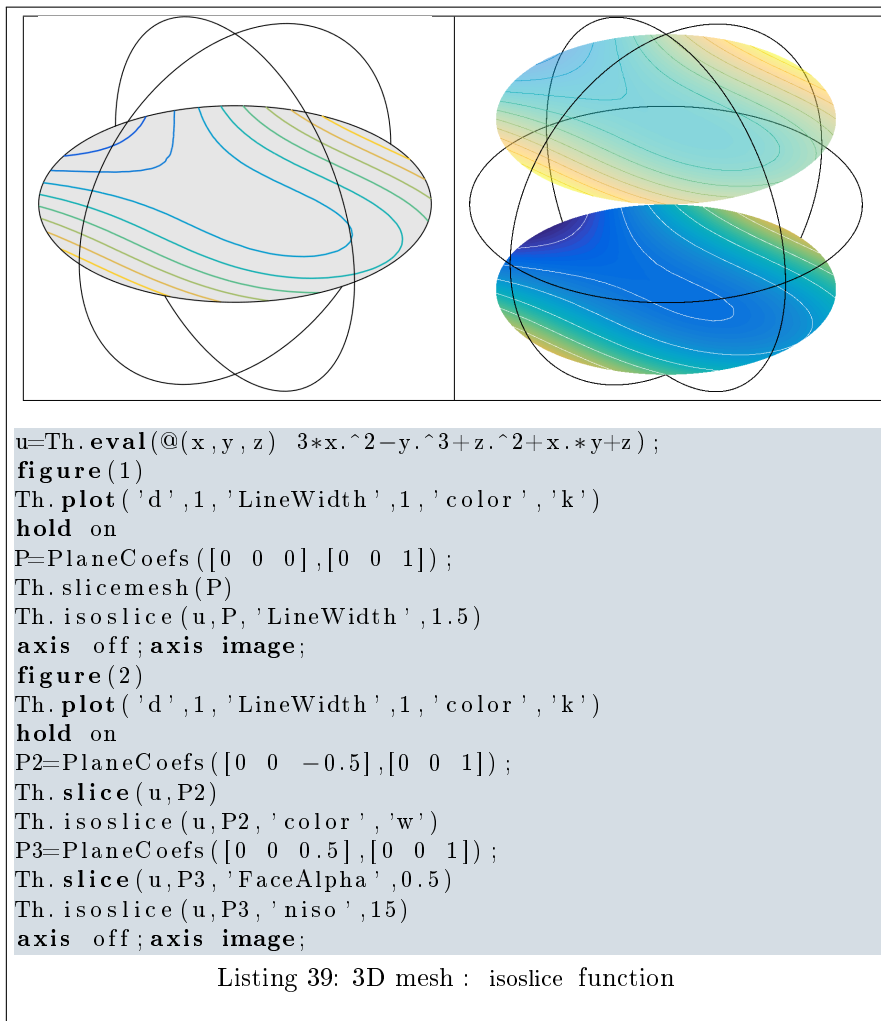
Description

`Th.isoslice(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. 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 `PlaneCoefs` of the `mooMesh` toolbox. With this function, the array `P`, is obtained with $P = \text{PlaneCoefs}(Q,V)$ where `Q` is a point in the plane and `V` is a vector orthogonal to it.

`Th.slide(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,
- `'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)
- `'mouse'` : if true, display information on clicked isoline (default : false)

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.



3.2.8 function plotVectorField

The method `plotVectorField` displays vector field datas on the mesh or parts of the mesh defined by an `ooMesh` object.

Syntaxe

```

Th.plotVectorField(V)
Th.plotVectorField(V,Name,Value,...)

```

Description

`Th.plotVectorField(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 .

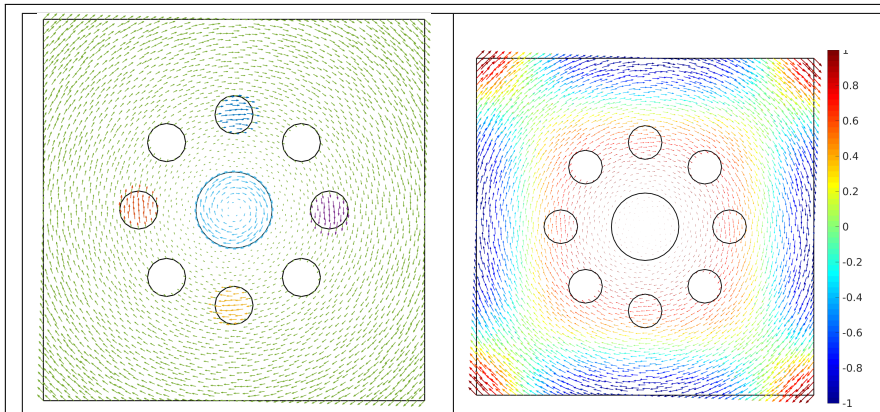
`Th.plotVectorField(V,Name,Value, ...)` specifies function options using one or more Name,Value pair arguments. Options of first level are

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

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`.

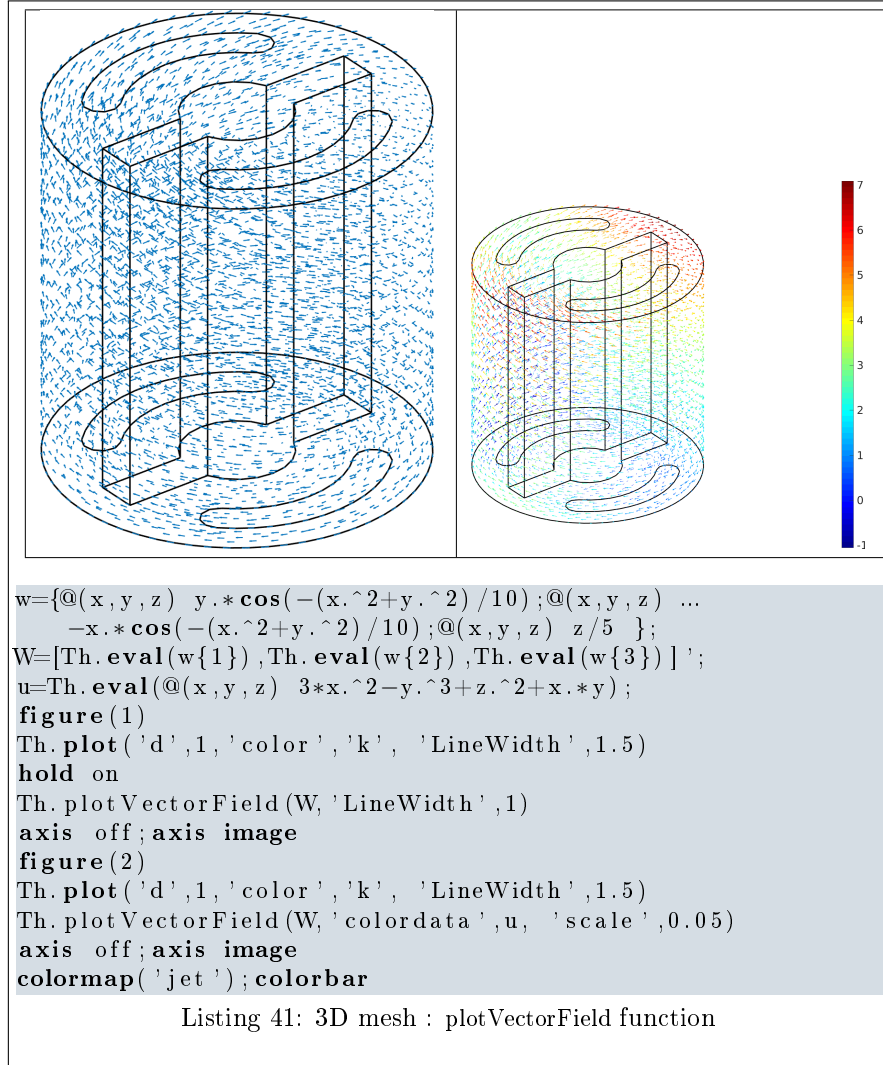
2D example The following example use the *.geo* file `condenser11.geo` which is in the directory `geodir` of the toolbox.



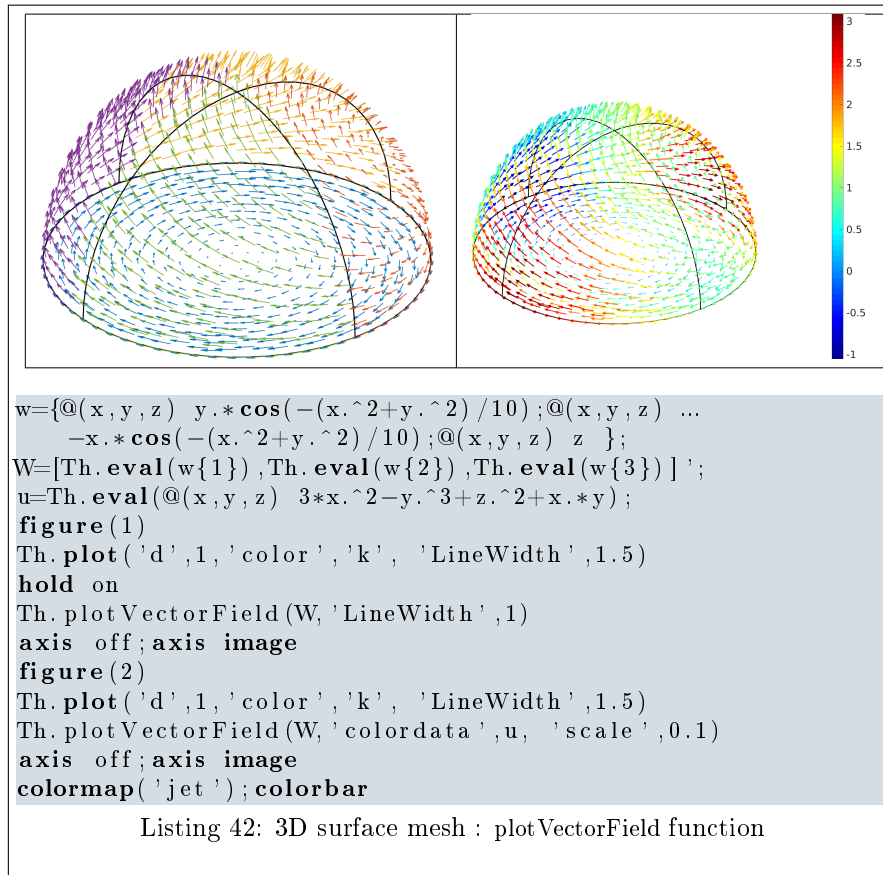
```
u=@(x,y) cos(pi*x.^2).*cos(pi*y.^2);
U=Th.eval(u);
w={@(x,y) y.*cos(-(x.^2+y.^2)/10);@(x,y) ...
    -x.*cos(-(x.^2+y.^2)/10)};
W=[Th.eval(w{1}),Th.eval(w{2})]';
figure(1)
Th.plot('d',1,'color','k','LineWidth',1.5)
hold on
Th.plotVectorField(W,'LineWidth',1)
axis off;axis image
figure(2)
Th.plot('d',1,'color','k','LineWidth',1.5)
Th.plotVectorField(W,'colordata',U,'scale',0.05)
axis off;axis image
colormap('jet');colorbar
```

Listing 40: 2D mesh : `plotVectorField` function

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.



3.3 ooMeshDD methods

3.3.1 ooMeshDD constructor

The constructor of the `ooMeshDD` class can initialize the object from a partitioned mesh file (obtained from `gmsh`) or from an `ooMesh` object.

remark 3.1

The interfaces between mesh partitions must have their labels less than 0 or, greater than or equal to 10000.

Syntaxe

```

ThDD=ooMeshDD(meshfile)

ThDD=ooMeshDD(Th)

```

Description

`ThDD=ooMeshDD(meshfile)` create the `ooMeshDD` object `ThDD` from the mesh file `meshfile` (gmsh format by default).

`ThDD=ooMeshDD(Th)` create the `ooMeshDD` object `ThDD` from the `ooMesh` object `Th`.

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

```
Matlab commands with output

meshfile=gmsh.buildmesh2d('condenser11',25,'verbose',0);
mpfile=gmsh.buildpartmesh2d(meshfile,5,'verbose',0);
disp('***_Read_partitioned_mesh_***')
ThDD=ooMeshDD(mpfile)

*** Read partitioned mesh ***

ThDD =

ooMeshDD with properties:

    d: 2
   dim: 2
  order: 1
    np: 5
    nq: 3474
     Th: {1x5 cell}
    bbox: [-1 1 -1 1]
 toGlobal: [1x3474 double]
  pLabels: [1 2 3 4 5]
  pColors: [5x3 double]
 PhysElts: {[1x1 struct] [1x1 struct]}
 InterElts: {[1x1 struct]}
```

3.3.2 function eval

The method `eval` evaluates a function on a partitioned mesh defined by an `ooMeshDD` object.

Syntaxe

```
Th.eval(fun)
Th.eval(fun,Name,Value,...)
```

Description

`Th.eval(fun)` evaluates the function `fun` on all the vertices of the partitioned mesh. The return value is an array of dimension `Th.nq-by-1`. If `fun` is a cell array of m functions, the return value is a `Th.nq-by- $m$` array.

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

- `'split'` : the return value is splitted by partitions and so its a cell array of dimension `Th.np` where each cell (one by partition) is an array of dimension the number of vertices of the partition by m .

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

```

                                Matlab commands with output

u=@(x,y) cos(x+y);
Th.info()
disp('without_splitting')
{Th.eval(u)}
disp('with_splitting')
Th.eval(u,'split',true)

Variable Th [ooMeshDD object] :
dim=2, d=2, np=7
nq=3474, nne=6690
Partition 1, label 1 :
nq=521, nne=955
Partition 2, label 2 :
nq=523, nne=956
Partition 3, label 3 :
nq=520, nne=956
Partition 4, label 4 :
nq=523, nne=955
Partition 5, label 5 :
nq=523, nne=956
Partition 6, label 6 :
nq=521, nne=956
Partition 7, label 7 :
nq=522, nne=956
without_splitting

ans =

    [3474x1 double]

with_splitting

ans =

    [521x1 double]
    [523x1 double]
    [520x1 double]
    [523x1 double]
    [523x1 double]
    [521x1 double]
    [522x1 double]

```

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.

Matlab commands with output

```
u=@(x,y,z) cos(x+y-z);
Th.info()
disp('***_without_splitting_***')
{Th.eval(u)}
disp('***_with_splitting_***')
Th.eval(u,'split',true)
```

```
Variable Th [ooMeshDD object] :
    dim=3, d=3, np=7
    nq=5296, nme=22526
Partition 1 , label 1 :
    nq=841, nme=3218
Partition 2 , label 2 :
    nq=834, nme=3218
Partition 3 , label 3 :
    nq=817, nme=3218
Partition 4 , label 4 :
    nq=855, nme=3218
Partition 5 , label 5 :
    nq=835, nme=3218
Partition 6 , label 6 :
    nq=857, nme=3218
Partition 7 , label 7 :
    nq=899, nme=3218
*** without splitting ***

ans =

    [5296x1 double]

*** with splitting ***

ans =

    [841x1 double]
    [834x1 double]
    [817x1 double]
    [855x1 double]
    [835x1 double]
    [857x1 double]
    [899x1 double]
```

3D surface 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.

Matlab commands with output

```
u=@(q) cos(q(1,:)+q(2,:)-q(3,:));
Th.info()
disp('***_without_splitting_***')
{Th.eval(u)}
disp('***_with_splitting_***')
Th.eval(u,'split',true)
```

```
Variable Th [ooMeshDD object] :
  dim=3, d=2, np=7
  nq=3344, nme=6688
Partition 1 , label 1 :
  nq=518, nme=955
Partition 2 , label 2 :
  nq=519, nme=957
Partition 3 , label 3 :
  nq=527, nme=955
Partition 4 , label 4 :
  nq=520, nme=956
Partition 5 , label 5 :
  nq=520, nme=954
Partition 6 , label 6 :
  nq=522, nme=955
Partition 7 , label 7 :
  nq=519, nme=956
*** without splitting ***

ans =

[3344x1 double]

*** with splitting ***

ans =

[518x1 double]
[519x1 double]
[527x1 double]
[520x1 double]
[520x1 double]
[522x1 double]
[519x1 double]
```

3.3.3 function plot

The method `plot` displays completes or partials partitions of a partitioned mesh defined by an `ooMeshDD` object.

Syntaxe

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

Description

`Th.plot()` displays all the `Th.d`-dimensional simplices elements of all the partitions.

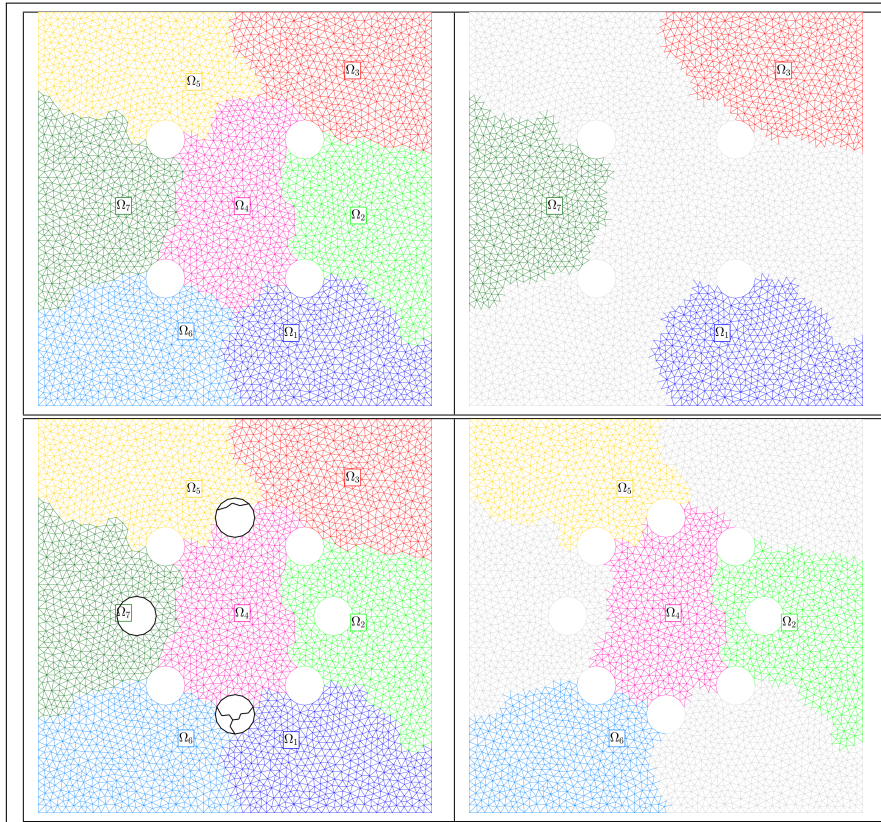
`Th.plot(Name,Value,...)` specifies function options using one or more `Name,Value` pair arguments. Options of first level are

- `'parts'` : to select the labels of the partitions to display,
- `'labels'` : to select the labels of the elements to display,
- `'color'` : to specify the color of the displayed partitions. (default : use one color by partition),

- **'legend'** : add a legend to graph if true (default : false)
- **'LegendOptions'** : options passed to legend (**text** function if Th.dim==2 and **legend** function if Th.dim==3)

The options of second level depend on the type of elementaries mesh elements to represent. (see **plot** function of **mooMesh** class)

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



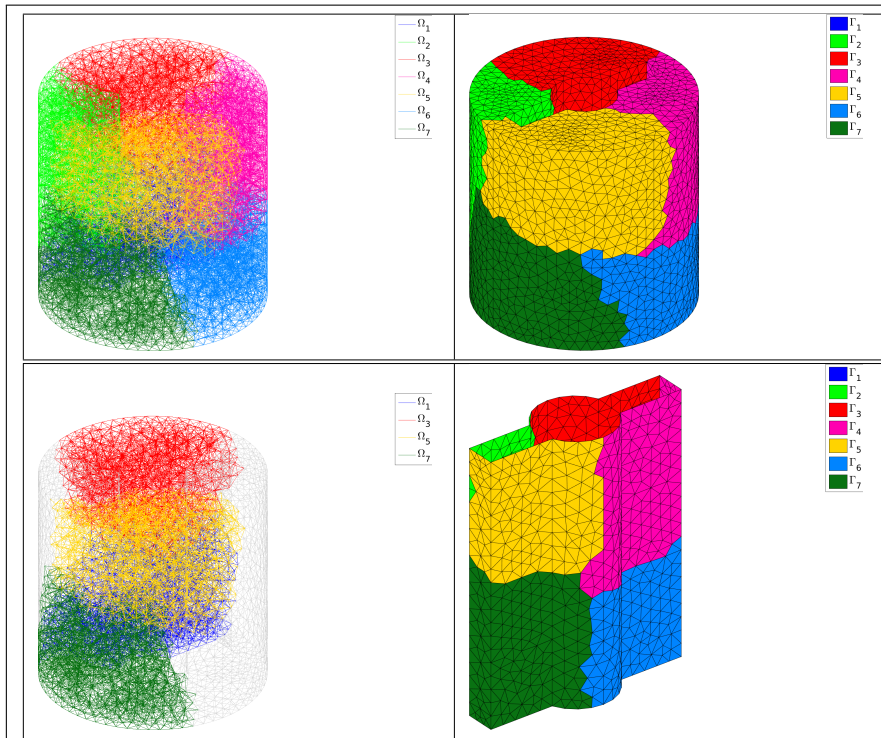
```

figure(1)
Th.plot('legend',true)
axis off;axis equal;
figure(2)
Th.plot('parts',[1,3,7],'legend',true)
hold on;axis off;axis equal
Th.plot('parts',[2,4,5,6],'color',0.8*[1,1,1])
figure(3)
Th.plot('labels',[105,110],'legend',true)
hold on;axis off;axis equal
Th.plot('bounds',true,'labels',[120:10:140],'color',[0 0 ...
    0],'LineWidth',1.5)
figure(4)
Th.plot('parts',[2,4,5,6], ...
    'labels',[105,110],'legend',true)
hold on;axis off;axis equal
Th.plot('parts',[1,3,7], ...
    'labels',[105,110],'color',0.8*[1,1,1])

```

Listing 43: Function lstinlineplot for 2D partitioned mesh

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.



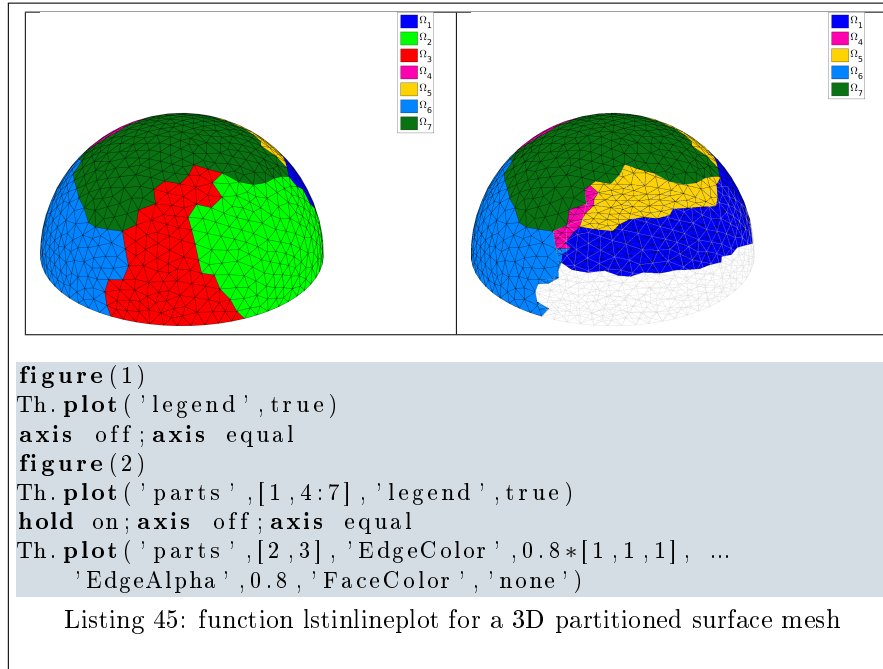
```

figure(1)
Th.plot('legend',true)
hold on
Th.plot('d',1,'Color','k','Linewidth',1.5)
axis off;axis equal
figure(2)
Th.plot('d',2,'legend',true)
view(3);hold on;axis off;axis equal
figure(3)
Th.plot('d',2,'labels',[1,1000,1020,1021], ...
        'EdgeColor',0.9*[1,1,1], ...
        'EdgeAlpha',0.4,'FaceColor','none')
hold on;axis off;axis equal
Th.plot('d',2,'labels',1000,'bounds',true,'color','k')
Th.plot('d',2,'labels',[10,11,31,2000,2020,2021])
figure(4)
Th.plot('d',2,'EdgeColor',0.9*[1,1,1], ...
        'EdgeAlpha',0.4,'FaceColor','none')
hold on;axis off;axis equal
Th.plot('d',1,'legend',true)

```

Listing 44: function lstinlineplot for a 3D partitioned mesh

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.



3.3.4 function `plotPhysicals`

The method `plotPhysicals` displays completes or partials physicals mesh elements of a partitioned mesh defined by an `ooMeshDD` object.

Syntaxe

```

Th.plotPhysicals()
Th.plotPhysicals(Name,Value, ...)

```

Description

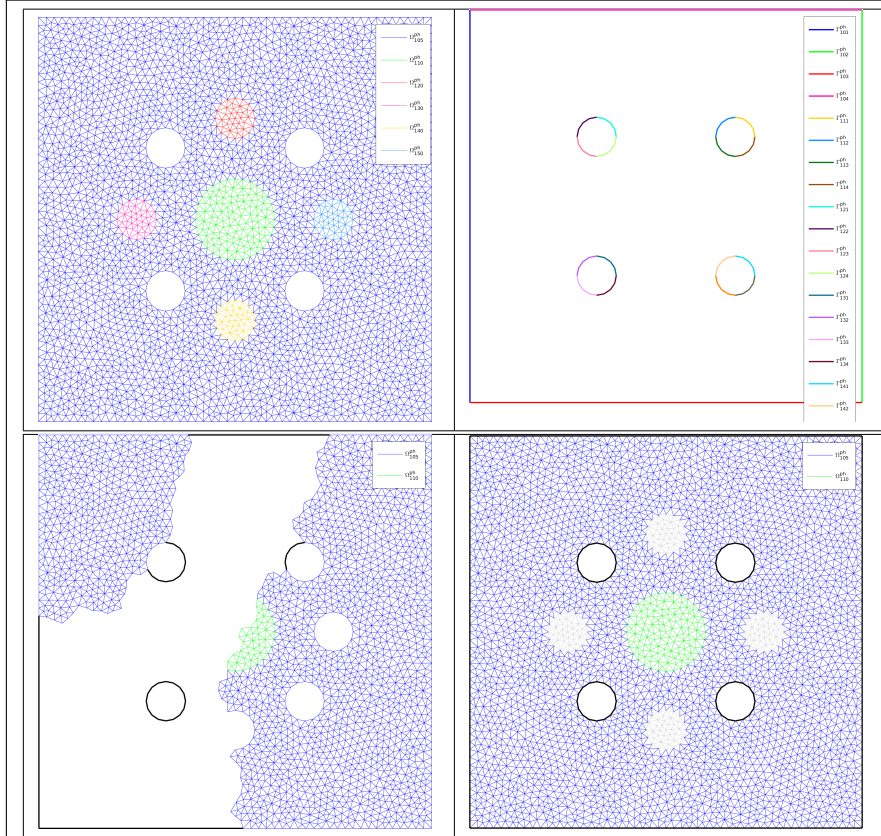
`Th.plotPhysicals()` displays all the `Th.d`-dimensional simplices physicals elements of the mesh.

`Th.plotPhysicals(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`)
- `'parts'` : to select the labels of the partitions on which to display physical elements,
- `'labels'` : to select the labels of the physical elements to display,
- `'color'` : to specify the color of the displayed physical elements. (default : use one color by physical element),
- `'legend'` : add a legend to graph if true (default : false)
- `'LegendOptions'` : options passed to `legend` function.

The options of second level depend on the type of elementaries mesh elements to represent. (see options of second level of the **plot** function of **mooMesh** class)

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



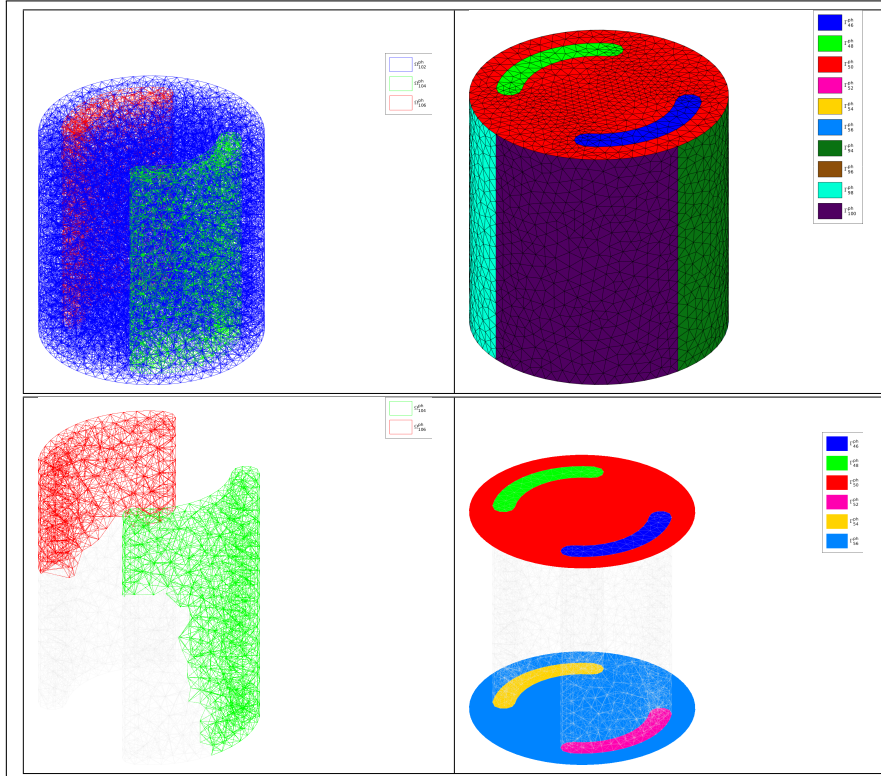
```

figure(1)
Th.plotPhysicals('legend',true)
axis off;axis equal;
figure(2)
Th.plotPhysicals('d',1,'legend',true,'LineWidth',2)
axis off;axis equal;
figure(3)
Th.plotPhysicals('parts',[1:4],'labels',[105,110],'legend',true)
hold on
Th.plotPhysicals('d',1,'parts',[5:7],'color','k','LineWidth',2)
figure(4)
Th.plotPhysicals('labels',[105,110],'legend',true)
hold on
Th.plotPhysicals('d',1,'color','k','LineWidth',2)
Th.plotPhysicals('labels',[120:10:150],'color',0.8*[1,1,1])

```

Listing 46: Function **plotPhysicals** on a 2D partitioned mesh

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.



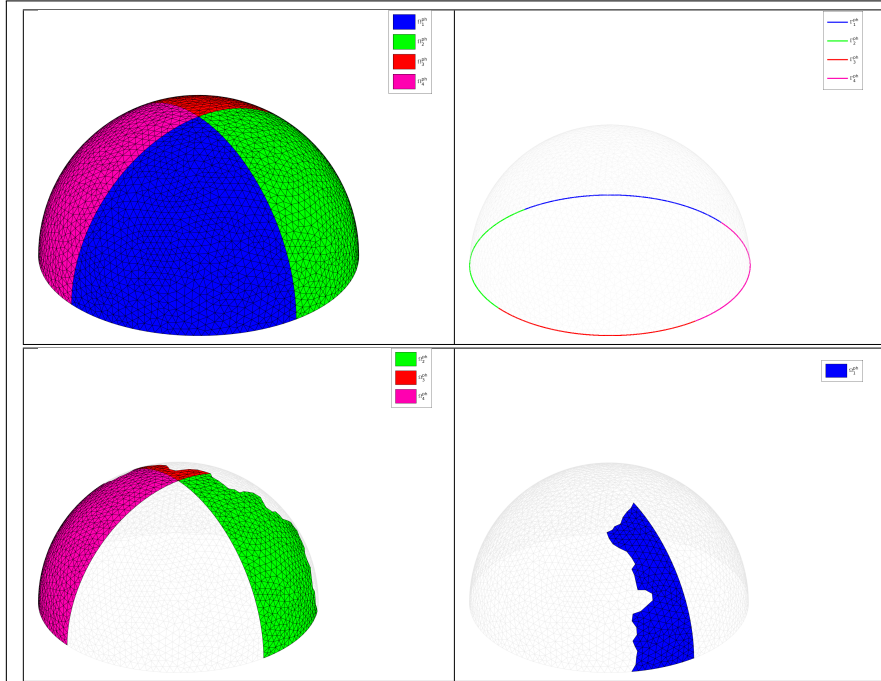
```

figure(1)
Th.plotPhysicals('legend',true)
axis off;axis equal
figure(2)
Th.plotPhysicals('d',2,'legend',true)
view(3);hold on;axis off;axis equal
figure(3)
Th.plotPhysicals('labels',[104,106],'parts',4:5, ...
    'color',0.9*[1,1,1],'EdgeAlpha',0.3)
view(3);hold on;axis off;axis equal
Th.plotPhysicals('labels',[104,106],'parts',1:3, ...
    'legend',true)
figure(4)
Th.plotPhysicals('labels',[104,106],'color',0.9*[1,1,1], ...
    'EdgeAlpha',0.3)
hold on;axis off;axis equal
Th.plotPhysicals('d',2,'legend',true,'labels',46:2:56, ...
    'EdgeColor','none')

```

Listing 47: Function `plotPhysicals` on 3D partitioned mesh

3D surface example The following example use the *.geo* file `demisphere4surf.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.



```

figure(1)
Th.plotPhysicals('legend',true)
axis off;axis equal
figure(2)
Th.plotPhysicals('FaceColor','none', ...
    'EdgeColor',0.9*[1,1,1],'EdgeAlpha',0.3);
hold on;axis off;axis equal
Th.plotPhysicals('d',1,'legend',true,'LineWidth',2)
figure(3)
Th.plotPhysicals('labels',1,'FaceColor','none', ...
    'EdgeColor',0.9*[1,1,1]);
Th.plotPhysicals('parts',1:2,'FaceColor','none', ...
    'EdgeColor',0.9*[1,1,1]);
hold on;axis off;axis equal
Th.plotPhysicals('labels',2:4,'parts',3:5,'legend',true)
figure(4)
Th.plotPhysicals('labels',2:4,'FaceColor','none', ...
    'EdgeColor',0.9*[1,1,1]);
Th.plotPhysicals('parts',4:5,'FaceColor','none', ...
    'EdgeColor',0.9*[1,1,1]);
hold on;axis off;axis equal
Th.plotPhysicals('labels',1,'parts',1:3,'legend',true)

```

Listing 48: Function `plotPhysicals` on a 3D surface partitioned mesh

3.3.5 function `plotInterfaces`

The method `plotInterfaces` displays completes or partials interfaces between partitions of a partitioned mesh defined by an `ooMeshDD` object.

Syntaxe

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

Description

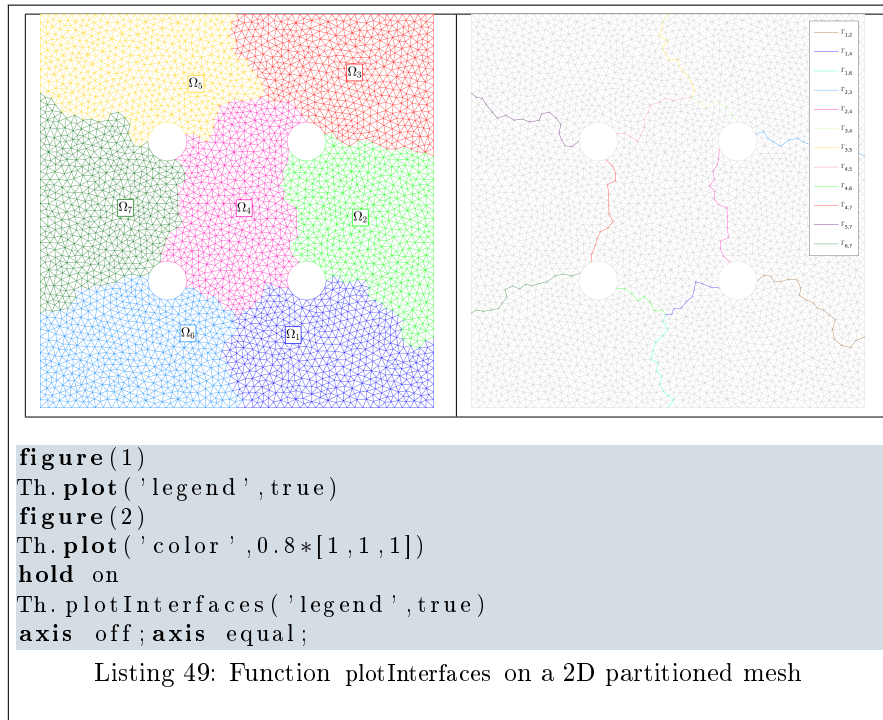
`Th.plotInterfaces()` displays all the interfaces between the partitions of a mesh. For a d -simplicial partitioned mesh, the interfaces are $(d - 1)$ -simplicial elementary meshes.

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

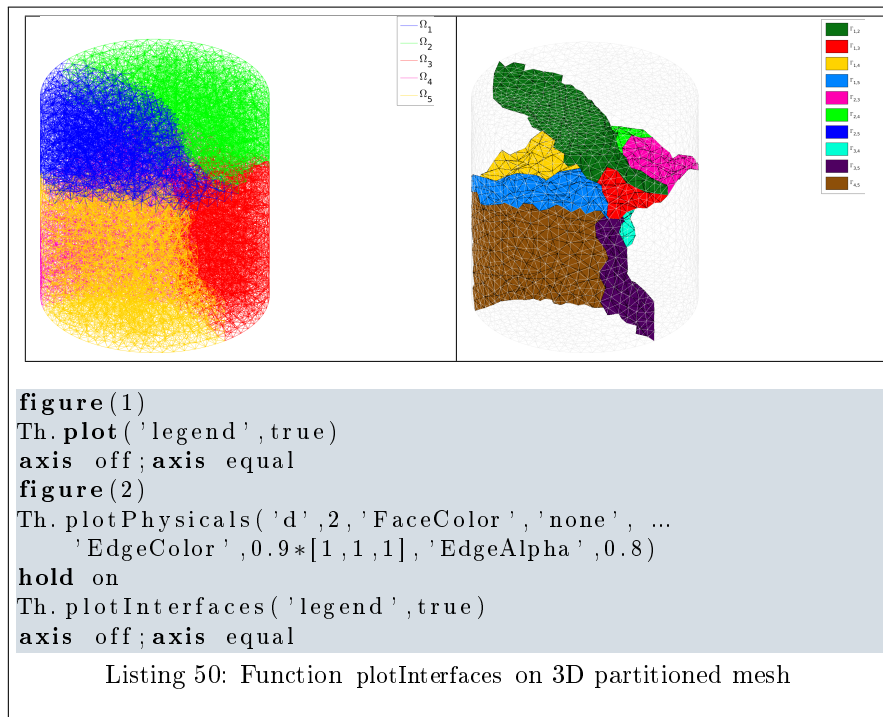
- `'labels'` : to select the labels of the physical elements to display,
- `'color'` : to specify the color of the displayed physical elements. (default : use one color by physical element),
- `'idx'` : indices of the interfaces.
- `'legend'` : add a legend to graph if true (default : false)
- `'LegendOptions'` : options passed to `legend` function.

The options of second level depend on the type of elementaries mesh elements to represent. (see options of second level of the `plot` function of `mooMesh` class)

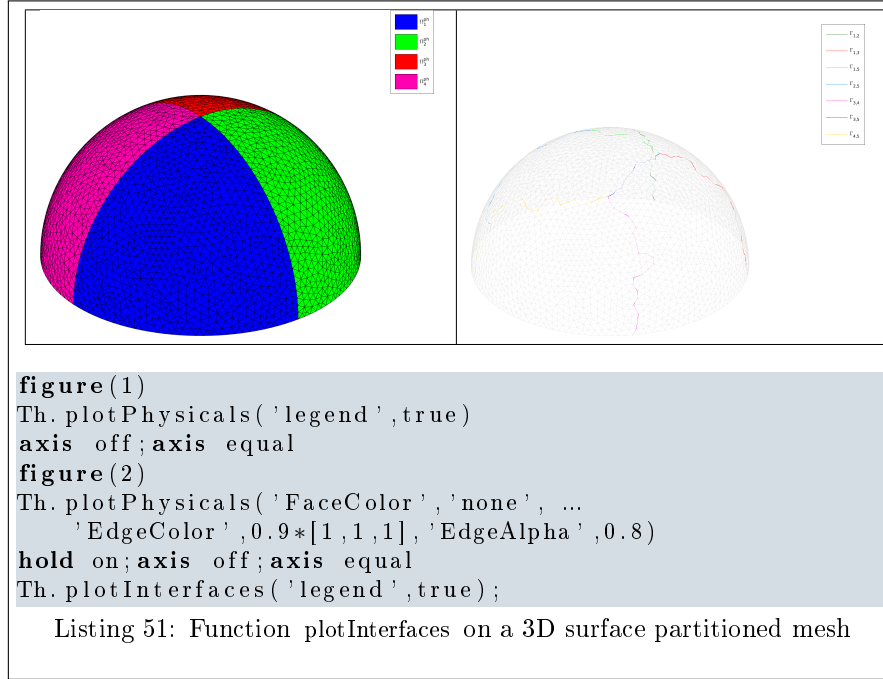
2D example The following example use the `.geo` file `condenser11.geo` which is in the directory `geodir` of the toolbox.



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 `demisphere4surf.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.



3.3.6 function `plotVal`

The method `plotVal` displays datas on the partitioned mesh or parts of the partitioned mesh defined by an `ooMeshDD` object.

Syntaxe

```

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

```

Description

`Th.plotVal(u)` displays data `u` on all the `Th.d`-dimensional simplices elements. The data `u` is either a cell array of size `Th.np` or an 1D-array of size `Th.nq` or `Th.nqGlobal` or `Th.nqParent`.

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

- **'parts'** : to select the labels of the partitions on which to display the data `u`,
- **'d'** : to specify the dimension of the simplices elements (default : `Th.d`)

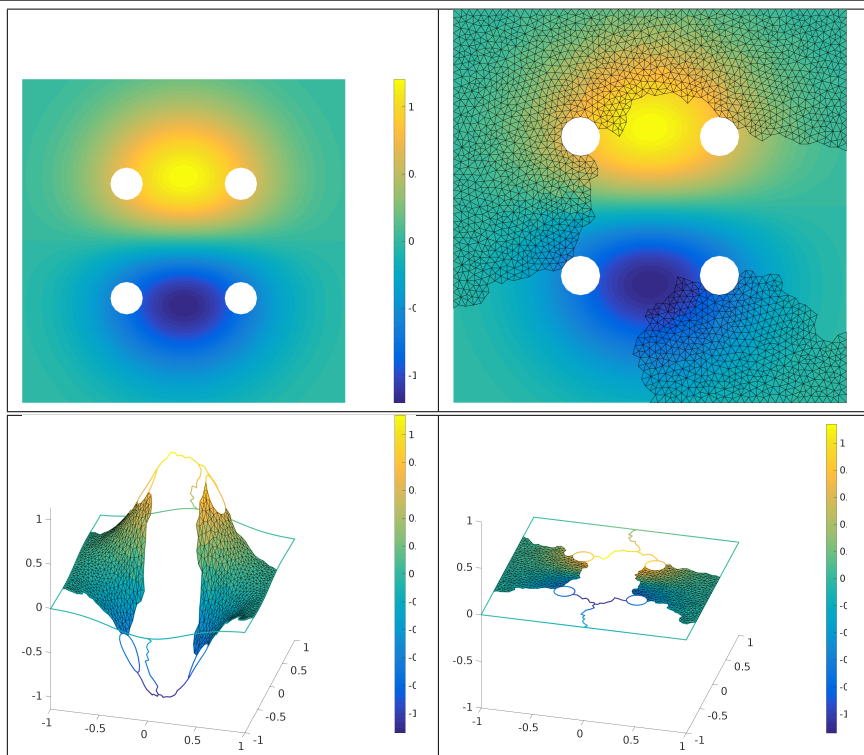
- **'labels'** : to select the labels of the elements on which to represent data,
- **'plan'** : if true, (default : false)

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 **'plan'** is true, **patch** function is used, otherwise **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.



```

u=Th.eval(@(x,y) 5*exp(-3*(x.^2+y.^2)).*cos(x).*sin(y));
figure(1)
Th.plotVal(u)
shading interp;axis off;axis equal;
colorbar
figure(2)
Th.plotVal(u,'parts',1:2:7)
hold on
Th.plotVal(u,'parts',[2,4,6],'FaceColor','interp',...
'EdgeColor','none')
axis off;axis equal;

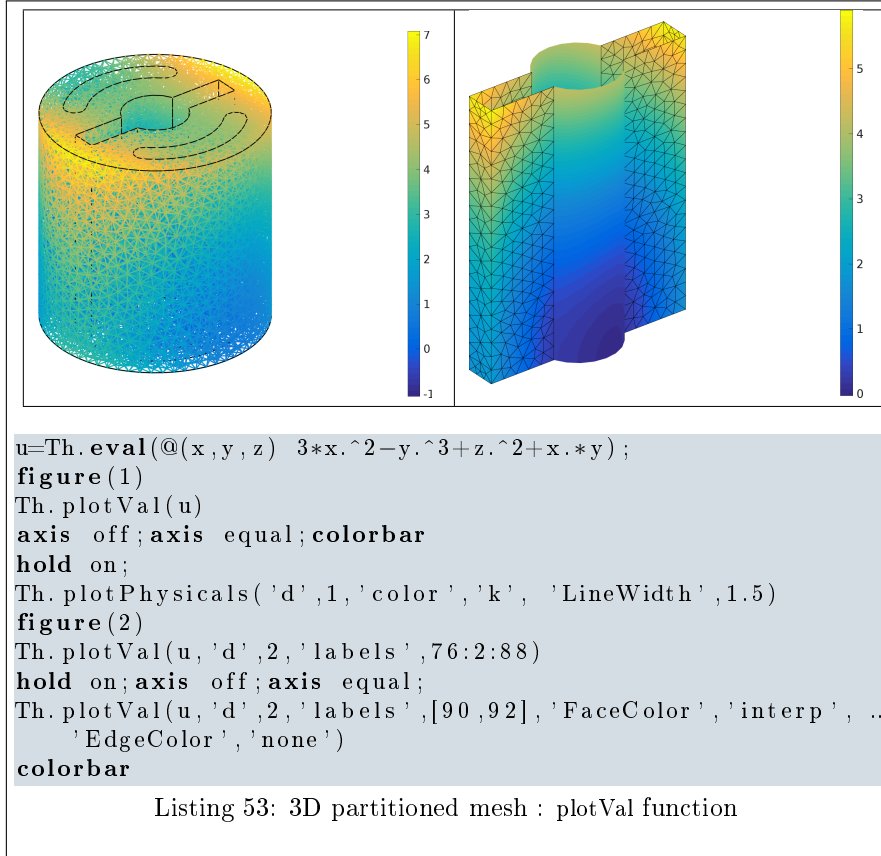
figure(3)
Th.plotVal(u,'d',1)
hold on
Th.plotVal(u,'parts',[2,7])
view(14.4,28.4);
axis equal;colorbar

figure(4)
Th.plotVal(u,'d',1,'plan',true)
hold on
Th.plotVal(u,'parts',[2,7],'plan',true)
view(14.4,28.4);
axis equal;colorbar

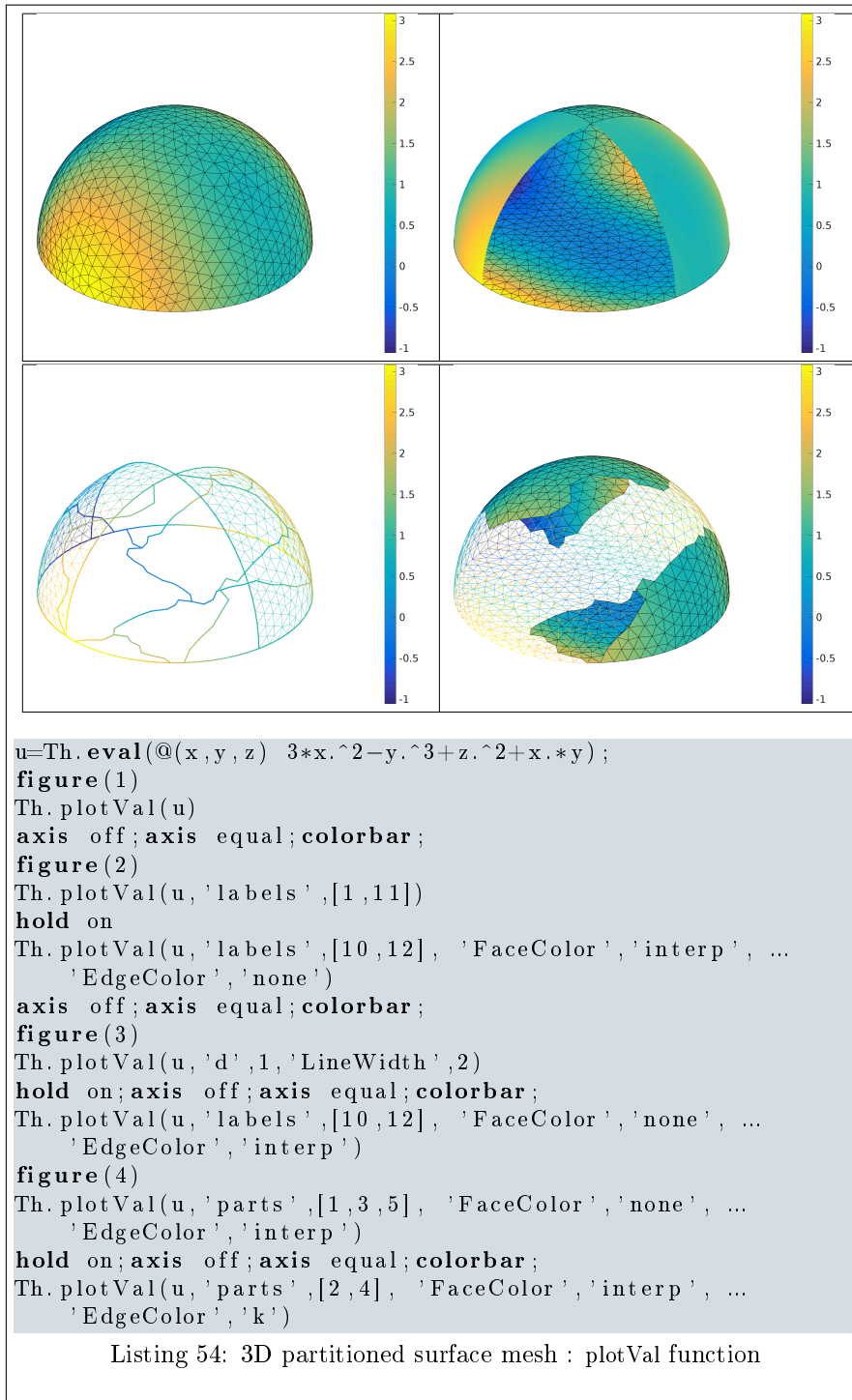
```

Listing 52: 2D partitioned mesh : plotValDD function

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.



3.3.7 function plotIsolines

The method `plotIsolines` displays isolines from datas on the partitioned mesh or parts of the partitioned mesh defined by an `ooMesh` object. This function

only works with 2-simplices in space dimension 2 or 3.

Syntaxe

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

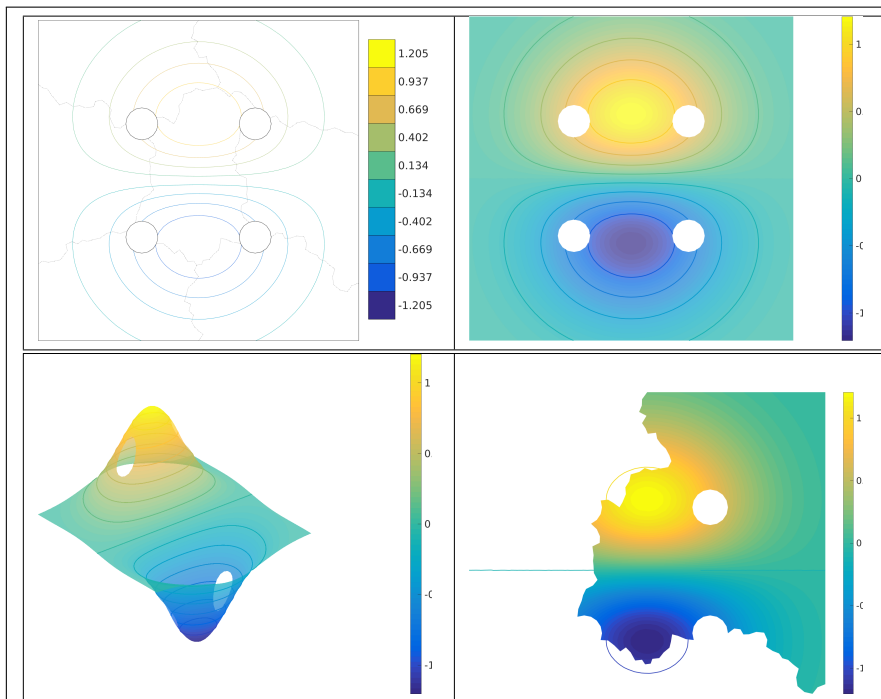
Description

See method `plotIsolines` of `ooMesh` class for description (see `??`).

A supplementary Name,Value pair was added :

- **'parts'** : to select the labels of the partitions on which to display isolines.

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



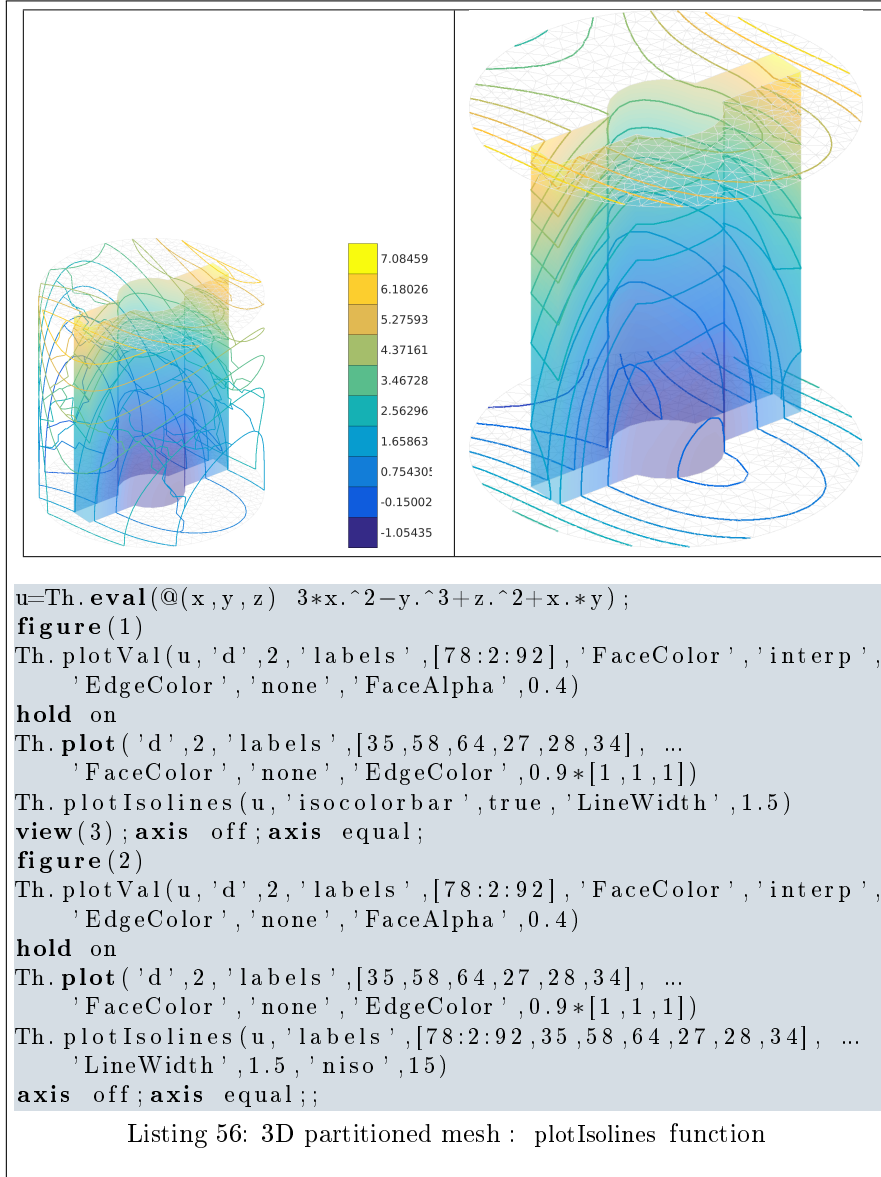
```

u=Th.eval(@(x,y) 5*exp(-3*(x.^2+y.^2)).*cos(x).*sin(y));
figure(1)
Th.plotPhysicals('d',1,'color','k')
hold on
Th.plotInterfaces('color','k','LineStyle',':')
Th.plotIsolines(u,'isocolorbar',true,'format','%3f')
axis off;axis equal;
figure(2)
Th.plotVal(u,'plan',true,'FaceAlpha',0.7)
shading interp;hold on
Th.plotIsolines(u,'plan',true,'LineWidth',1.5)
axis off;axis equal;colorbar
figure(3)
Th.plotVal(u,'FaceAlpha',0.7)
view(3)
shading interp;hold on
Th.plotIsolines(u,'niso',15,'LineWidth',1.5)
axis off;axis equal;
colorbar
figure(4)
Th.plotVal(u,'plan',true,'parts',[2,3,4])
shading interp;hold on
Th.plotIsolines(u,'isorange',0,'LineWidth',1.5,...
'color','w')
Th.plotIsolines(u,'isorange',[-1,1], 'LineWidth',1.5,...
'color','k','plan',true)
axis off;axis equal;colorbar

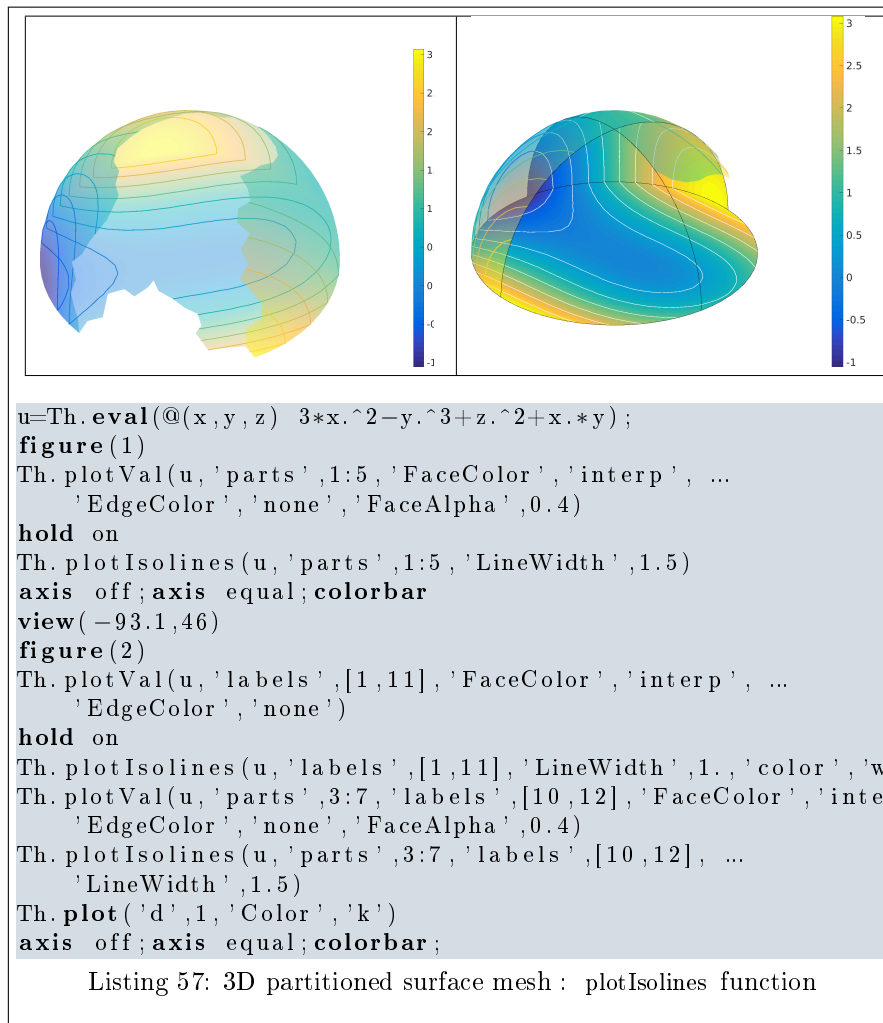
```

Listing 55: 2D partitioned mesh : plotIsolines function

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.



3.3.8 function plotVectorField

The method `plotVectorField` displays vector field datas on the mesh or parts of the partitioned mesh defined by an `ooMeshDD` object.

Syntaxe

```

Th.plotVectorField(V)
Th.plotVectorField(V,Name,Value,...)

```

Description

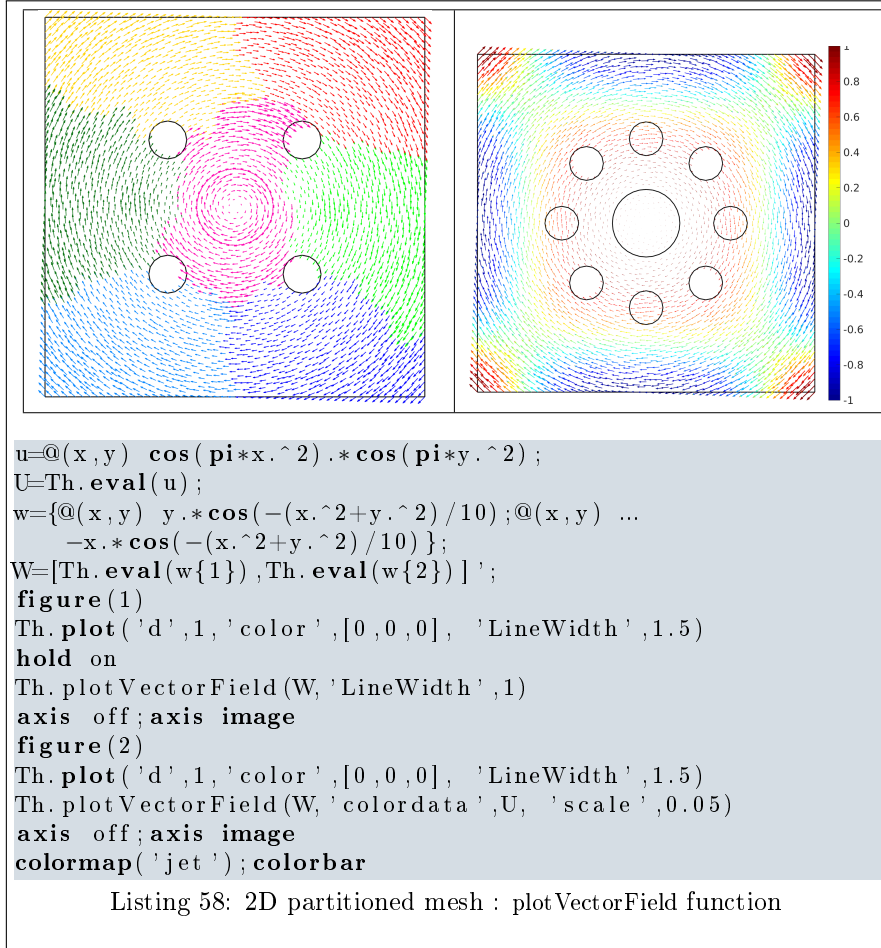
See method `plotVectorField` of `ooMesh` class for description (see ??).

A supplementary Name,Value pair was added :

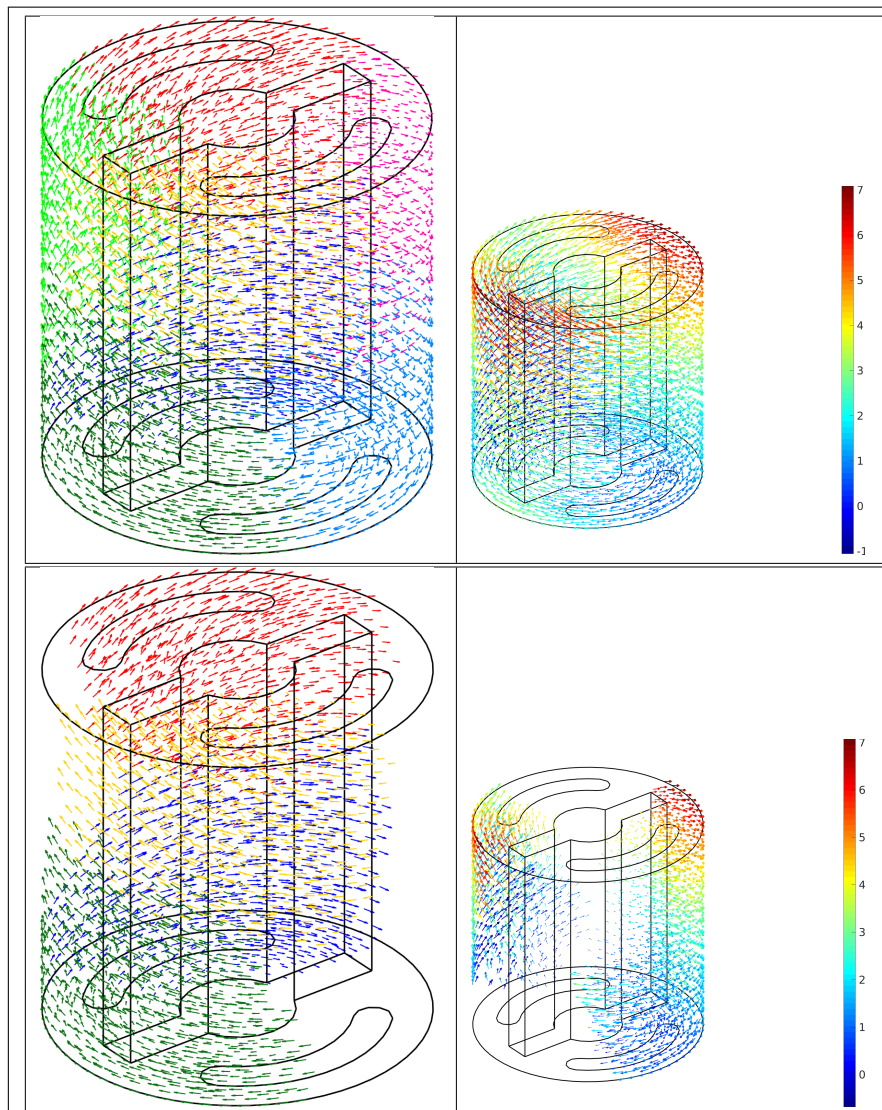
- `'parts'` : to select the labels of the partitions on which to display isolines.

The 'colordata' option accept also, as value, a cell array of size Th.np.

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



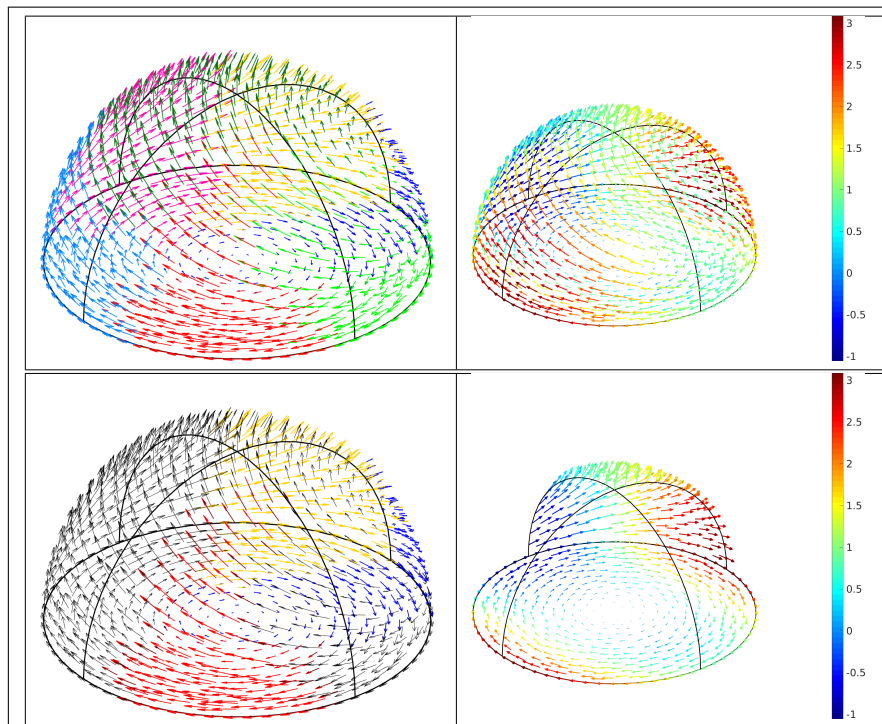
3D example The following example use the .geo file `cylinderkey.geo` which is in the directory `geodir` of the toolbox. This file contains description of a 3D mesh with simplices of dimensions 1, 2 and 3.



```
w=@(x,y,z) y.*cos(-(x.^2+y.^2)/10);@(x,y,z) ...
    -x.*cos(-(x.^2+y.^2)/10);@(x,y,z) z/5 };
W=[Th.eval(w{1}),Th.eval(w{2}),Th.eval(w{3})]';
u=Th.eval(@(x,y,z) 3*x.^2-y.^3+z.^2+x.*y);
figure(1)
Th.plot('d',1,'color','k','LineWidth',1.5)
hold on
Th.plotVectorField(W,'LineWidth',1)
axis off;axis image
figure(2)
Th.plot('d',1,'color','k','LineWidth',1.5)
Th.plotVectorField(W,'colordata',u,'scale',0.1)
axis off;axis image
colormap('jet');colorbar
figure(3)
Th.plot('d',1,'color','k','LineWidth',1.5)
hold on
Th.plotVectorField(W,'LineWidth',1,'parts',1:2:Th.np)
axis off;axis image
figure(4)
Th.plot('d',1,'color','k','LineWidth',1.5)
Th.plotVectorField(W,'colordata',u,...
    'scale',0.1,'parts',2:2:Th.np)
axis off;axis image;colormap('jet');colorbar
```

Listing 59: 3D partitioned mesh : plotVectorField function

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.



```
w=@(x,y,z) y.*cos(-(x.^2+y.^2)/10);@(x,y,z) ...
    -x.*cos(-(x.^2+y.^2)/10);@(x,y,z) z };
W=[Th.eval(w{1}),Th.eval(w{2}),Th.eval(w{3})]';
u=Th.eval(@(x,y,z) 3*x.^2-y.^3+z.^2+x.*y);
figure(1)
Th.plot('d',1,'color','k','LineWidth',1.5)
hold on
Th.plotVectorField(W,'LineWidth',1)
axis off;axis image
figure(2)
Th.plot('d',1,'color','k','LineWidth',1.5)
Th.plotVectorField(W,'colordata',u,'scale',0.1)
axis off;axis image
colormap('jet');colorbar
figure(3)
Th.plot('d',1,'color','k','LineWidth',1.5)
hold on
Th.plotVectorField(W,'LineWidth',1,'parts',[1,3,5])
Th.plotVectorField(W,'color','k','parts',[2,4,6,7])
axis off;axis image
figure(4)
Th.plot('d',1,'color','k','LineWidth',1.5)
Th.plotVectorField(W,'colordata',u,...
    'scale',0.1,'labels',[1,11])
axis off;axis image
colormap('jet');colorbar
```

Listing 60: 3D partitioned surface mesh : plotVectorField function

3.3.9 function slice

The method `slice` displays datas on the intersection of a plane and a 3D partitioned mesh defined by an `ooMesh` object.

Syntaxe

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

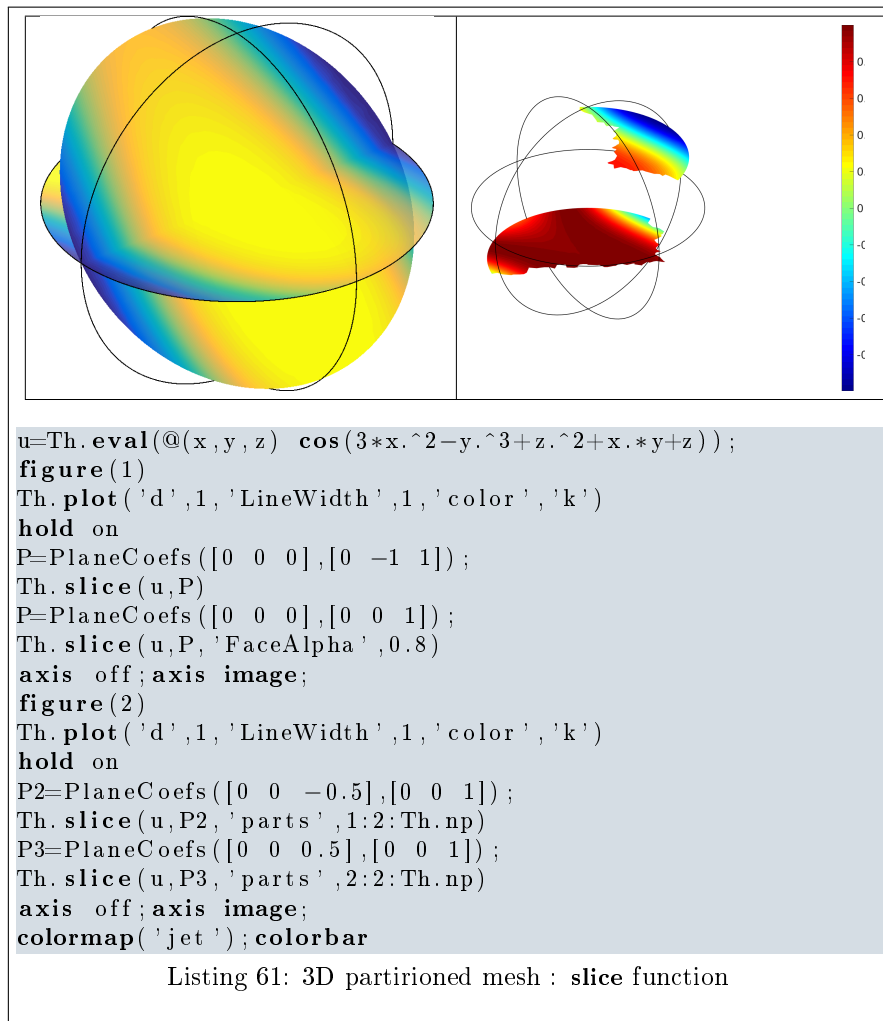
Description

See method `slice` of `ooMesh` class for description (see ??).

A supplementary Name,Value pair was added :

- **'parts'** : to select the labels of the partitions on which to display the slice.

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.



3.3.10 function slicemesh

The method **slicemesh** displays intersection of a plane and a 3D partitioned mesh defined by an **ooMesh** object.

Syntaxe

```

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

```

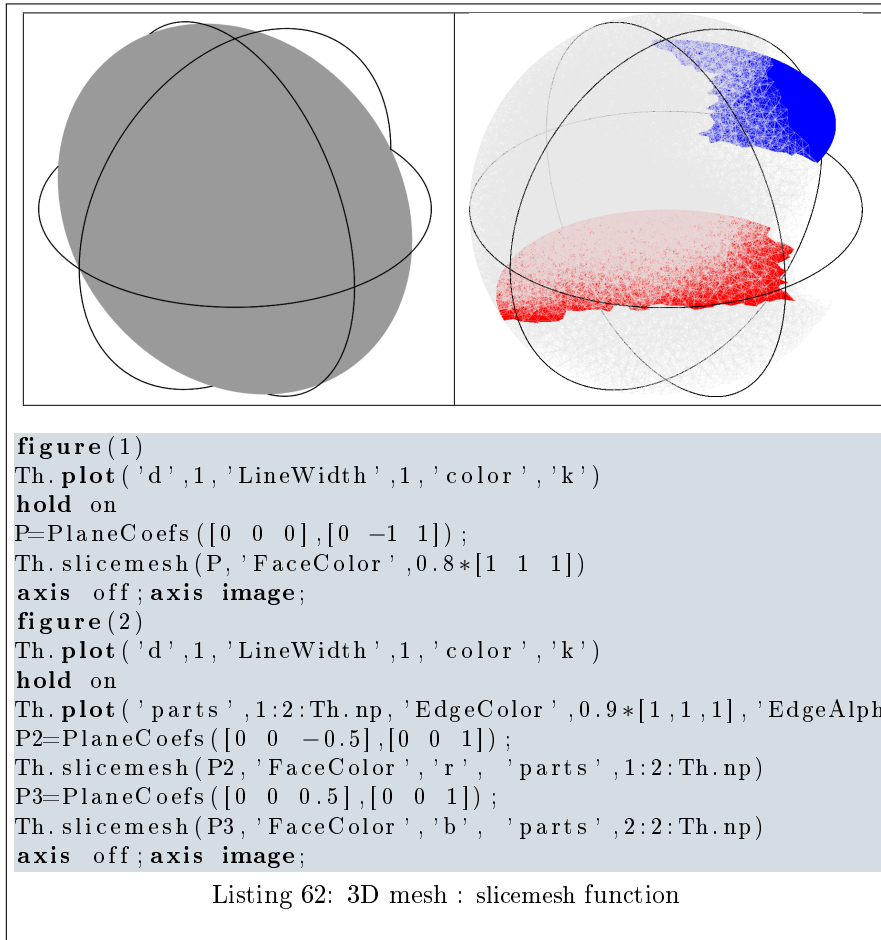
Description

See method **slicemesh** of **ooMesh** class for description (see ??).

A supplementary Name,Value pair was added :

- **'parts'** : to select the labels of the partitions on which to display the slice.

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.



3.3.11 function isoslice

The method `isoslice` displays datas as isolines on the intersection of a plane and a 3D partitioned mesh defined by an `ooMesh` object.

Syntaxe

```

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

```

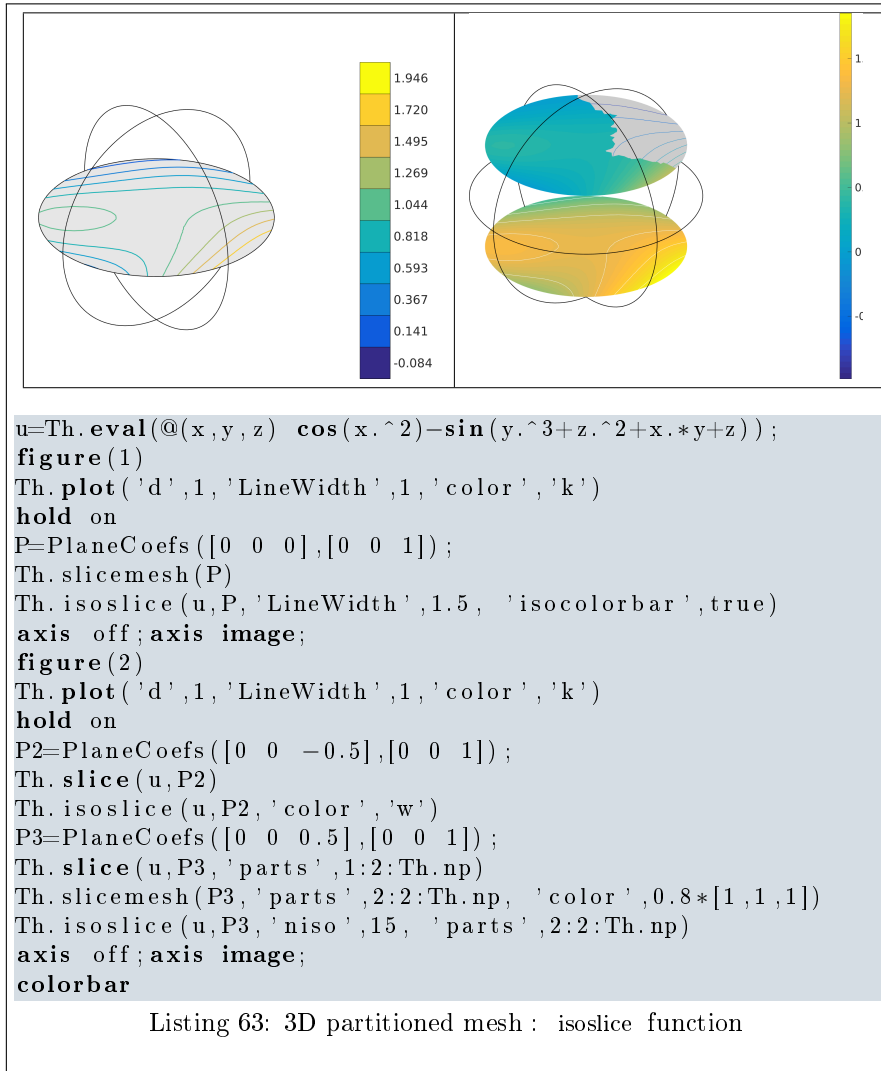
Description

See method `isoslice` of `ooMesh` class for description (see ??).

A supplementary Name,Value pair was added :

- **'parts'** : to select the labels of the partitions on which to display the slice.

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.



3.3.12 function plotInterfacesVal

The method **plotInterfacesVal** displays datas on interfaces between partitions of a partitioned mesh defined by an **ooMeshDD** object.

Syntaxe

```
Th.plotInterfacesVal(xi)
Th.plotInterfaces(xi,Name,Value, ...)
```

Description

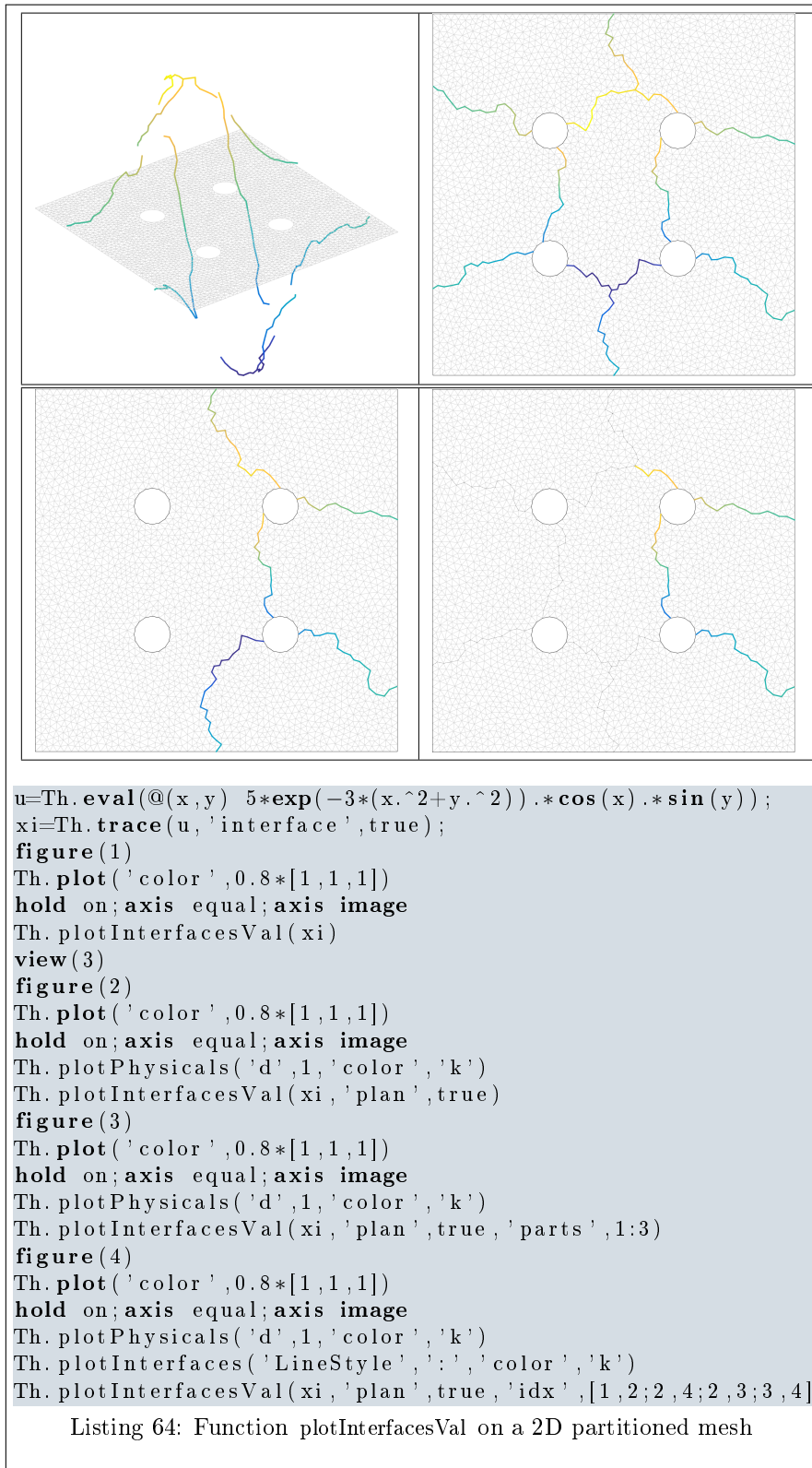
`Th.plotInterfacesVal(xi)` displays datas on all the interfaces between the partitions of a mesh. For a d -simplicial partitioned mesh, the interfaces are $(d - 1)$ -simplicial elementary meshes. The data `xi` is a N -by- N cell array where N is the number of partitions. `xi(i,j)` is an array.

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

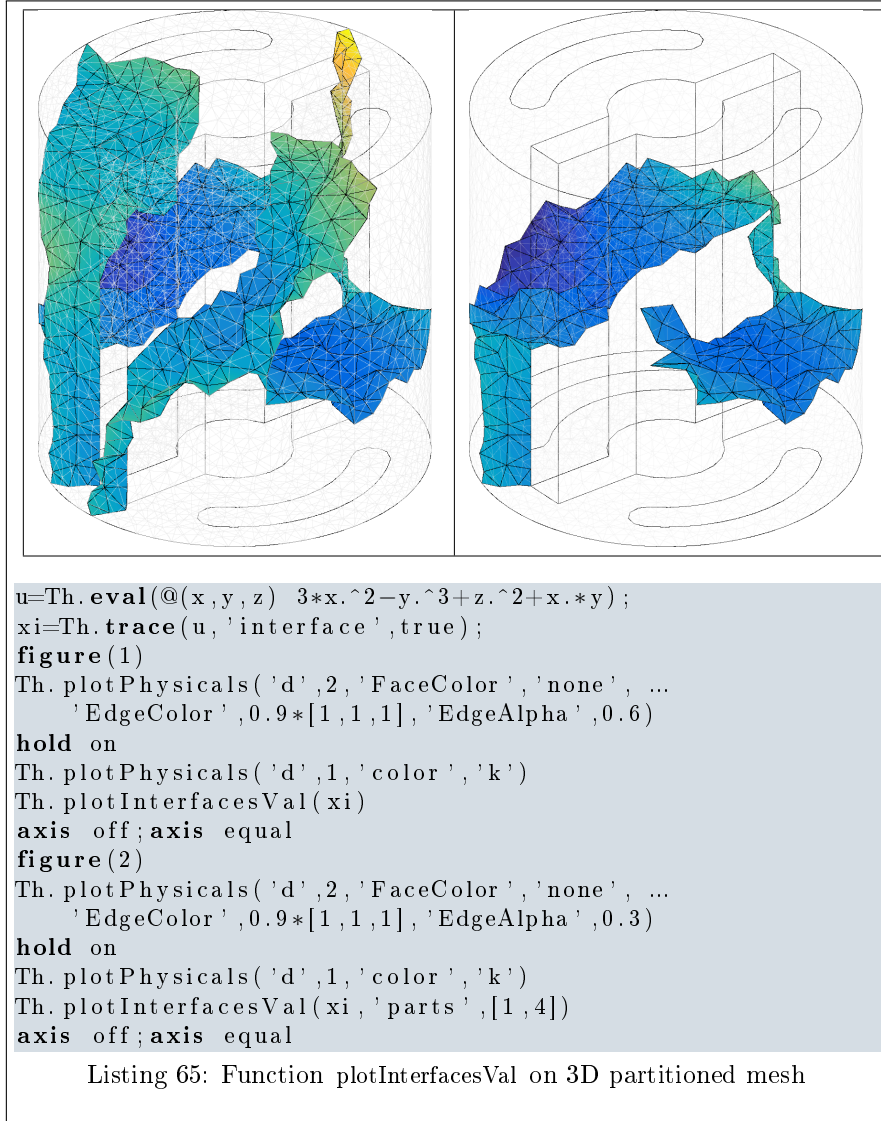
- **'parts'** : to select the labels of the partitions,
- **'idx'** : indices of the interfaces.
- **'upper'** : if true (default) display datas `xi(i,j)` on interface $\Gamma_{i,j}$ for $i > j$ otherwise its for $i < j$.
- **'plan'** : only for 2D meshes. if true, force displaying in 2D (default : false)

The options of second level depend on the type of elementaries mesh elements to represent. (see options of second level of the `plotVal` function of `mooMesh` class)

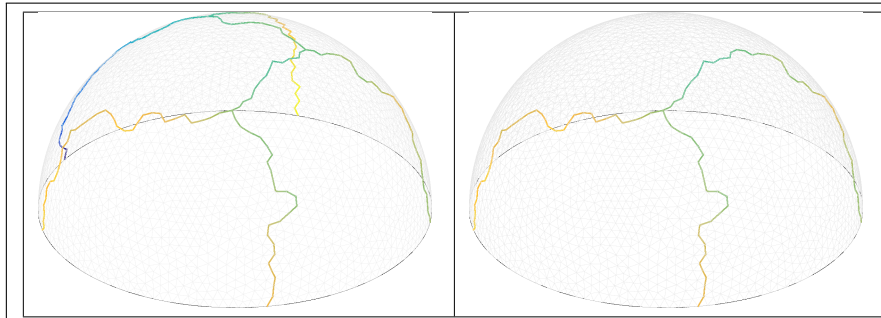
2D example The following example use the `.geo` file `condenser11.geo` which is in the directory `geodir` of the toolbox.



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 `demisphere4surf.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.



```

u=Th.eval(@(x,y,z) 3*x.^2-y.^3+z.^2+x.*y);
xi=Th.trace(u,'interface',true);
figure(1)
Th.plotInterfacesVal(xi,'LineWidth',2)
hold on
Th.plotPhysicals('d',1,'color','k')
Th.plotPhysicals('FaceColor','none',...
    'EdgeColor',0.9*[1,1,1],'EdgeAlpha',0.6)
axis off;axis equal
figure(2)
Th.plotPhysicals('FaceColor','none',...
    'EdgeColor',0.9*[1,1,1],'EdgeAlpha',0.8)
Th.plotPhysicals('d',1,'color','k')
hold on;axis off;axis equal
Th.plotInterfacesVal(xi,'LineWidth',2,'idx',[1,3;3,5;4,3;5,4])

```

Listing 66: Function `plotInterfacesVal` on a 3D surface partitioned mesh

3.3.13 function `plotInterfacesIsolines`

The method `plotInterfacesIsolines` displays isolines on interfaces between partitions of a 3D partitioned mesh defined by an `ooMeshDD` object.

Syntaxe

```

Th.plotInterfacesIsolines(xi)
Th.plotInterfacesIsolines(xi,Name,Value,...)

```

Description

`Th.plotInterfacesIsolines(xi)` displays isolines data on all the interfaces between the partitions of a mesh. For a d -simplicial partitioned mesh, the interfaces are $(d-1)$ -simplicial elementary meshes. The data `xi` is a N -by- N cell array where N is the number of partitions. `xi(i,j)` is an array.

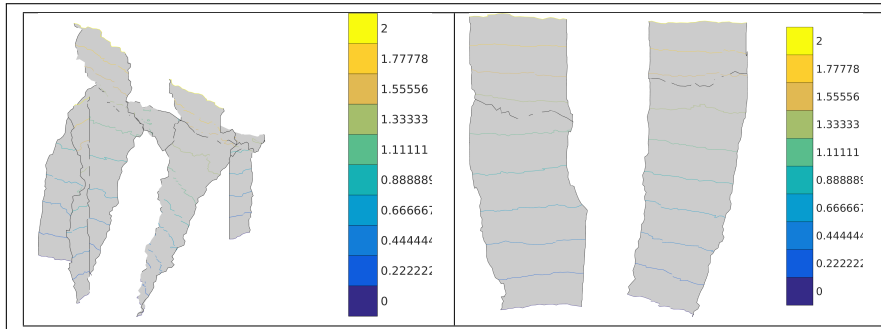
`Th.plotInterfacesIsolines(xi,Name,Value,...)` specifies function options using one or more `Name,Value` pair arguments. Options of first level are

- `'parts'` : to select the labels of the partitions,
- `'idx'` : indices of the interfaces.

- **'upper'** : if true (default) display datas $xi(i,j)$ on interface $\Gamma_{i,j}$ for $i > j$ otherwise its for $i < j$.
- **'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')

The options of second level depend on the type of elementaries mesh elements to represent. (see options of second level of the `plotIsolines` function of `mooMesh` class)

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.



```
u=Th.eval(@(x,y,z) z);
xi=Th.trace(u,'interface',true);
figure(1)
Th.plotInterfaces('FaceColor',0.8*[1,1,1],'EdgeColor','none')
hold on
Th.plotInterfaces('bounds',true,'color','k')
Th.plotInterfacesIsolines(xi,'isocolorbar',true)
axis off;axis equal
figure(2)
Th.plotInterfaces('FaceColor',0.8*[1,1,1],'EdgeColor','none', ...
'idx',[2,3;1,2;3,4;4,5])
hold on
Th.plotInterfaces('bounds',true,'color','k', ...
'idx',[2,3;1,2;3,4;4,5])
Th.plotInterfacesIsolines(xi,'isocolorbar',true, ...
'idx',[2,3;1,2;3,4;4,5])
axis off;axis equal;view(-82.3,18)
```

Listing 67: Function `plotInterfacesIsolines` on 3D partitioned mesh

3.4 ooMesh representations

3.5 Datas representations on Meshes

A Calcul des gradients des coordonnées barycentriques

Soit K un d -simplex dans $\mathbb{R}^m$ de sommets $q^1, \dots, q^{d+1}$ et $\hat{K}$ le d -simplex de référence dans $\mathbb{R}^d$ de sommets $\hat{q}^1, \dots, \hat{q}^{d+1}$ avec $\hat{q}^1 = \mathbf{0}_d$ et $\hat{q}^{i+1} = \mathbf{e}_i, \forall i \in \llbracket 1, d \rrbracket$.

We denote by $\mathcal{F}$ the linear map from $\hat{K}$ to K given by

$$q = \mathcal{F}_K(\hat{q}) = \mathbb{A}_K \hat{q} + q^1 \quad (1)$$

where $\mathbb{A}_K \in \mathcal{M}_{m,d}(\mathbb{R})$ is defined by

$$\mathbb{A}_K = \begin{pmatrix} q^2 - q^1 & \cdots & q^{d+1} - q^1 \end{pmatrix} \quad (2)$$

On note $\mathbb{H}_K \in \mathcal{M}_{d,d}(\mathbb{R})$ la matrice symétrique donnée par $\mathbb{H}_K = \mathbb{A}_K^t \mathbb{A}_K$. Elle est définie positive (et donc inversible) car le d -simplex K est non dégénéré. De (1) on déduit

$$\mathbb{A}_K^t (q - q^1) = \mathbb{A}_K^t \mathbb{A}_K \hat{q} = \mathbb{H}_K \hat{q}$$

et donc en notant $\mathbb{B}_K = \mathbb{H}_K^{-1} \mathbb{A}_K^t \in \mathcal{M}_{d,m}(\mathbb{R})$ on a

$$\hat{q} = \mathcal{F}_K^{-1}(q) = \mathbb{B}_K (q - q^1).$$

We can note that if $d = m$ then $\mathbb{B}_K = \mathbb{A}_K^{-1}$.

The barycentric coordinates of $\hat{q} = (\hat{x}_1, \dots, \hat{x}_d) \in \hat{K}$ are given by $\hat{\lambda}_0 = 1 - \sum_{i=1}^d \hat{x}_i$, and $\hat{\lambda}_{i+1} = \hat{x}_i, \forall i \in \llbracket 1, d \rrbracket$. They are polynomials of degree 1 and thus their derivatives are constants. The barycentric coordinates of $q = (x_1, \dots, x_m) \in K$ are given $\forall i \in \llbracket 1, d+1 \rrbracket$ by $\lambda_i(q) = \hat{\lambda}_i \circ \mathcal{F}_K^{-1}(q)$ and we have

$$\nabla \lambda_i(q) = \mathbb{B}_K^t \hat{\nabla} \hat{\lambda}_i(\hat{q}), \quad \forall i \in \llbracket 1, d+1 \rrbracket \quad (3)$$

with $\hat{\nabla} \hat{\lambda}_1(\hat{q}) = \begin{pmatrix} -1 \\ \vdots \\ -1 \end{pmatrix}, \hat{\nabla} \hat{\lambda}_{i+1} = \mathbf{e}_i, \forall i \in \llbracket 1, d \rrbracket$.

Let $\hat{\mathbb{G}} \in \mathcal{M}_{d,d+1}(\mathbb{R})$ and $\mathbb{G}_K \in \mathcal{M}_{m,d+1}(\mathbb{R})$ the *gradients* matrices given by

$$\hat{\mathbb{G}} = \begin{pmatrix} \hat{\nabla} \hat{\lambda}_0 & \cdots & \hat{\nabla} \hat{\lambda}_{d+1} \end{pmatrix} = \begin{pmatrix} -1 & 1 & 0 & \cdots & 0 \\ -1 & 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ -1 & 0 & \cdots & 0 & 1 \end{pmatrix}$$

and

$$\mathbb{G}_K = \begin{pmatrix} \nabla \lambda_0 & \cdots & \nabla \lambda_{d+1} \end{pmatrix}$$

From (3), we have

$$\mathbb{G}_K = \mathbb{B}_K^t \hat{\mathbb{G}} = \mathbb{A}_K \mathbb{H}_K^{-1} \hat{\mathbb{G}}$$

Algorithm 1 gradients basis functions on a d -simplex in dimension $m \geq d$.

Input $\mathbf{q}$: m -by- $(d+1)$ array, $\mathbf{q}(:,i)$ is the i -th vertex of the d -simplex

Output $\mathbb{G}_K$: gradients array (m -by- $(d+1)$)
 $\mathbb{G}_K(:,\alpha) = \nabla \lambda_\alpha(\mathbf{q}), \forall \alpha \in \llbracket 1, d+1 \rrbracket$

Function $\mathbf{G} \leftarrow \text{GRADIENTS}(\mathbf{q})$

$\hat{\mathbb{G}} \leftarrow [-\mathbf{1}_d, \mathbb{I}_{d \times d}]$ $\triangleright$ Contains gradients of $\hat{\lambda}_i$

$\mathbb{A}_K \leftarrow \mathbf{q}(:, 2:d+1)^t - \mathbf{1}_d * \mathbf{q}(:, 1)^t$

$\mathbb{H}_K \leftarrow \mathbb{A}_K^t * \mathbb{A}_K$

$\mathbb{G}_K \leftarrow \mathbb{A}_K * \text{INV}(\mathbb{H}_K) * \hat{\mathbb{G}}$

end Function

Let $k \in \llbracket 1, n_{me} \rrbracket$. The k -th simplex of $\mathcal{T}_h$ is denoted by $K = T_k$. To simplify the notations we identify $\mathbb{A}_k = \mathbb{A}_{T_k}$, $\mathbb{B}_k = \mathbb{B}_{T_k}$, and so on. We want to compute all the gradients of $d+1$ barycentric coordinates on T_k denoted by $\lambda_0^k, \dots, \lambda_d^k$.

With these notations, we have

$$\mathbb{G}_k \stackrel{\text{def}}{=} \begin{pmatrix} \nabla \lambda_0^k & \vdots & \nabla \lambda_d^k \end{pmatrix} = \mathbb{B}_k^t \hat{\mathbb{G}} = \mathbb{A}_k \mathbb{H}_k^{-1} \hat{\mathbb{G}}, \quad \forall k \in \llbracket 1, n_{me} \rrbracket. \quad (4)$$

To vectorize the computation of all these gradients, we can rewrite in an equivalent way the n_{me} equations (4)

Let $\mathbb{A}$ be the block matrix defined by

$$\mathbb{A} \stackrel{\text{def}}{=} \begin{pmatrix} \mathbb{A}_1 & \mathbb{O} & \dots & \mathbb{O} \\ \mathbb{O} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbb{O} \\ \mathbb{O} & \dots & \mathbb{O} & \mathbb{A}_{n_{me}} \end{pmatrix}_{(m \times n_{me}) \times (d \times n_{me})} \quad (5)$$

where each block is a $m \times d$ matrix. So $\mathbb{A}$ is a $(m \times n_{me}) \times (d \times n_{me})$ matrix.

Let $N = d \times n_{me}$ and $\mathbb{H} = \mathbb{A}^t \mathbb{A}$. So the block matrix $\mathbb{H}$ is the $(d \times n_{me}) \times (d \times n_{me})$ matrix given by

$$\mathbb{H} \stackrel{\text{def}}{=} \begin{pmatrix} \mathbb{H}_1 & \mathbb{O} & \dots & \mathbb{O} \\ \mathbb{O} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbb{O} \\ \mathbb{O} & \dots & \mathbb{O} & \mathbb{H}_{n_{me}} \end{pmatrix}_{N \times N} \quad (6)$$

The n_{me} equations (4) can be solved in two steps :

1. Find $(\mathbb{X}_1, \dots, \mathbb{X}_{n_{me}})$ solution of

$$\begin{pmatrix} \mathbb{H}_1 & \mathbb{O} & \dots & \mathbb{O} \\ \mathbb{O} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbb{O} \\ \mathbb{O} & \dots & \mathbb{O} & \mathbb{H}_{n_{me}} \end{pmatrix}_{N \times N} \begin{pmatrix} \mathbb{X}_1 \\ \mathbb{X}_2 \\ \vdots \\ \mathbb{X}_{n_{me}} \end{pmatrix}_{N \times (d+1)} = \begin{pmatrix} \mathbb{A}_1^t \hat{\mathbb{G}} \\ \mathbb{A}_2^t \hat{\mathbb{G}} \\ \vdots \\ \mathbb{A}_{n_{me}}^t \hat{\mathbb{G}} \end{pmatrix}_{N \times (d+1)} \quad (7)$$

2. Compute

$$\begin{pmatrix} \mathbb{G}_1 \\ \mathbb{G}_2 \\ \vdots \\ \mathbb{G}_{n_{me}} \end{pmatrix}_{(m \times n_{me}) \times (d+1)} = \mathbb{A} \begin{pmatrix} \mathbb{X}_1 \\ \mathbb{X}_2 \\ \vdots \\ \mathbb{X}_{n_{me}} \end{pmatrix}_{N \times (d+1)} \quad (8)$$

B gmshoptions

Print.GeoOnlyPhysicals

Geometry.LabelType

Mesh.LabelType Type of element label (0=element number, 1=elementary entity number, 2=physical entity number, 3=partition number, 4=coordinates). *Default value: 0.*

Mesh.MetisAlgorithm Metispartitioning algorithm (1=Recursive, 2=K-way, 3=Nodal weight). *Default value: 1. Saved in: General.OptionsFileName*

Mesh.MetisEdgeMatching (Adv. **Metis**): Determines the matching type (1=Random, 2=Heavy-Edge, 3=Sorted Heavy-Edge) *Default value: 3 Saved in: General.OptionsFileName*

Mesh.MetisRefinementAlgorithm (Adv. **Metis**): Algorithm for k-way refinement (1=Random, 2=Greedy, 3=Random with minimized connectivity) *Default value: 3 Saved in: General.OptionsFileName*

Mesh.MshFilePartitioned Split MSH file by mesh partition (0: no, 1: yes, 2: create physicals by partition) *Default value: 0 Saved in: General.OptionsFileName*

Mesh.NbPartitions Number of partitions *Default value: 1 Saved in: General.OptionsFileName*

Mesh.Partitioner Partitioner software (1=Chaco, 2=**Metis**) *Default value: 2 Saved in: General.OptionsFileName*

Mesh.NbPartitions Number of partitions *Default value: 1 Saved in: General.OptionsFileName*

Mesh.SaveAll Ignore Physical definitions and save all elements *Default value: 0 Saved in: -*

Mesh.SaveElementTagType Type of the element tag saved in mesh formats that don't support saving physical or partition ids (1=elementary, 2=physical, 3=partition) *Default value: 1 Saved in: General.OptionsFileName*

Mesh.ZoneDefinition Method for defining a zone (0=single zone, 1=by partition, 2=by physical) *Default value: 0 Saved in: General.OptionsFileName*