fc_hypermesh Matlab toolbox, User's Guide *

François Cuvelier† Gilles Scarella ‡

June 29, 2018

Abstract

This object-oriented Matlab toolbox allows to generate conforming meshes of hypercubes, hyperrectangles or of any d-orthotopes by simplices or orthotopes with their m-faces. It was created to show the implementation of the algorithms of [?]. The fc_hypermesh toolbox uses Matlab objects and is provided with meshes visualisation tools for dimension less than or equal to 3.

0 Contents

*Compiled with Matlab 2017a, toolboxes **fc_hypermesh**-0.0.8 and **fc_tools**-0.0.23 under cosmos computer

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

‡Université Côte d'Azur, CNRS, LJAD, F-06108 Nice, France, gilles.scarella@unice.fr.

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

1 Introduction

The fc-hypermesh toolbox contains a simple class object OrthMesh which permits, in any dimension $d \geqslant 1$, to obtain a simplicial mesh or orthotope mesh with all their m-faces, $0 \leqslant m < d$. It is also possible with the method function `plotmesh` of the class object OrthMesh to represent a mesh or its m-faces for $d \leqslant 3$.

This toolbox was tested under

Windows 10.0.16299: with Matlab R2015b to R2018a

macOS High Sierra 10.13.4: with Matlab R2015b to R2018a

Ubuntu 16.04.3 LTS: with Matlab R2015b to R2018a

Ubuntu 17.10: with Matlab R2015b to R2018a

centOS 7.4: with Matlab R2015b to R2018a

Fedora 27: with Matlab R2015b to R2018a

OpenSUSE Leap 42.3: with Matlab R2015b to R2018a

With Matlab R2015b there is a trouble with the legend of the 0-faces (points) : same color for all points!

In the following section, the class object OrthMesh is presented. Thereafter some warning statements on the memory used by these objects in high dimension are given. Finally computation times for orthotope meshes and simplicial meshes are given in dimension $d \in [\![1, 5]\!]$.

2 Installation

2.1 Installation automatic, all in one (recommended)

For this method, one just has to get/download the install file

mfc_hypermesh_install.m

or get it on the dedicated web page. Thereafter, it should be run under Matlab. This command downloads, extracts and configures the *fc-hypermesh* and the required *fc-tools* toolboxes in the current directory.

For example, to install this toolbox in `~/Matlab/toolboxes` directory, one has to copy the file mfc_hypermesh_install.m in the `~/Matlab/toolboxes` directory. Then in a Matlab terminal run the following commands

```
>> cd ~/Matlab/toolboxes
>> mfc_hypermesh_install
```

This is the output of the `mfc_hypermesh_install` command on a Linux computer:

```
Parts of the <fc-hypermesh> Matlab toolbox.
Copyright (C) 2017-2018 F. Cuvelier <cuvelier@math.univ-paris13.fr>

1- Downloading and extracting the toolboxes
2- Setting the <fc-hypermesh> toolbox
Write in ~/Matlab/toolboxes/fc-hypermesh-full/fc_hypermesh-0.0.8/
     configure_loc.m ...
3- Using toolboxes :
   ->            fc-tools : 0.0.23
   ->        fc-hypermesh : 0.0.8
*** Using instructions
   To use the <fc-hypermesh> toolbox:
   addpath('~/Matlab/toolboxes/fc-hypermesh-full/fc_hypermesh-0.0.8')
   fc_hypermesh.init()

   See ~/Matlab/toolboxes/mfc_hypermesh_set.m
```

The complete toolbox (i.e. with all the other needed toolboxes) is stored in the directory `~/Matlab/toolboxes/fc-hypermesh-full` and, for each Matlab session, one has to set the toolbox by:

```
>> addpath('~/Matlab/toolboxes/fc-hypermesh-full/mfc-hypermesh-0.0.8')
>> fc_hypermesh.init()
```

To **uninstall**, one just has to delete directory

`~/Matlab/toolboxes/fc-hypermesh-full`

2.2 Manual installation

This package uses the `fc_tools` toolbox. So one has to install it as explain in the dedicated web page.

Thereafter, on the `fc_hypermesh` dedicated web page, one can found link to archives (*zip*, *7z* or *tar.gz* format)

- Downloads an archive and extracts it on a folder, for example `~/Matlab/toolboxes`. The toolbox path is `~/Matlab/toolboxes/mfc-hypermesh-0.0.8`
- Adds the toolbox path in Matlab with `addpath` command.
- Verifies that the the `fc_tools` toolbox is in the Matlab path. Otherwise, adds it...

3 Using the fc hypermesh toolbox

First of all, the main class object `OrthMesh` is presented. Thereafter some usage samples are given.

3.1 Class object OrthMesh

The aim of the class object OrthMesh is to efficiently create an object which contains a mesh of a d-orthotope and all its m-face meshes. An elementary mesh class object EltMesh is used to store only one mesh, the main mesh as well as any of the m-face meshes. This class EltMesh also simplifies (for me) the codes writing and its fields are the following:

- d, space dimension
- m, kind of mesh ($\mathtt{m} = \mathtt{d}$ for the main mesh)
- type, 0 for simplicial mesh or 1 for orthotope mesh
- nq, number of vertices
- q, vertices array of dimension d-by-nq
- nme, number of mesh elements
- me, connectivity array of dimension $(\mathtt{d}+1)$-by-nme for simplices elements or $2^{\mathtt{d}}$-by-nme for orthotopes elements
- toGlobal, index array linking local array q to the one of the main mesh
- label, name/number of this elementary mesh
- color, color of this elementary mesh (for plotting purpose)

Let the d-orthotope defined by $[a_1, b_1] \times \cdots \times [a_{\mathtt{d}}, b_{\mathtt{d}}]$. The class object OrthMesh corresponding to this d-orthotope contains the main mesh and all its m-face meshes, $0 \leqslant \mathtt{m} < \mathtt{d}$. Its Fields are the following

- d: space dimension
- type: string 'simplicial' or 'orthotope' mesh
- Mesh: main mesh as an EltMesh object
- Faces: list of arrays of EltMesh objects such that Faces(1) is an array of all the $(\mathtt{d}-1)$-face meshes, Faces(2) is an array of all the $(\mathtt{d}-2)$-face meshes, and so on
- box: a d-by-2 array such that box(i,1) $= a_i$ and box(i,2) $= b_i$.

3.1.1 Constructor

```
Oh = OrthMesh(d,N)
Oh = OrthMesh(d,N, key,value, ...)
```

Description

`Oh = OrthMesh(d,N)`

Genrerates the `OrthMesh` object `Oh` which contains which contains a simplicial mesh of the unit `d`-orthotope and all its `m`-face meshes.

`Oh = OrthMesh(d,N, key,value, ...)`

Some optional `key`/`value` pairs arguments are available with `key`:

- `'type'` : used to select the kind of elements used for meshing. The default `value` is `'simplicial'` and otherwise `'orthotope'` can be used.

  `Oh = OrthMesh(3,10, 'type','orthotope')`

- `'box'` : used to specify the `d`-orthotope $[a_1, b_1] \times \ldots \times [a_{\mathtt{d}}, b_{\mathtt{d}}]$ by setting `value` as an `d`-by-2 array such that a_i = `value(i,1)` and b_i = `value(i,2)`.

  `Oh = OrthMesh(3,10, 'box',[-1 1;-2 2;0 3])`

- `'m_min'` : used to only mesh the `m`-Faces for `m` in $[\![\mathtt{m}, \mathtt{d}]\!]$. Default `value` is 0.

  `Oh = OrthMesh(3,10, 'm_min',2)`

- `'mapping'` : used to apply on the mesh a mapping function given by a function handle.

  `Oh = OrthMesh(3,10, 'mapping',@(q) [q(1,:)+sin(q(2,:));q(2,:);q(3,:)])`

3.1.2 `plotmesh` method

The `plotmesh()` member function can be used to represent the mesh given by an `OrthMesh` object if the space dimension is less than or equal to 3.

Syntaxe

```
obj.plotmesh()
obj.plotmesh(key, value, ...)
```

Description

`obj.plotmesh()`

`obj.plotmesh(key, value, ...)`

Some optional `key`/`value` pairs arguments are available with `key`:

- `'legend'` : if `value` is `True`, a legend is displayed. Default is `False`.

- `'m'` : plots all the `m`-faces of the mesh. Default `m = d` i.e. the main mesh. ($0 \leqslant$ `m` $\leqslant$ `d`)
- `'color'` : use to specify the color of the mesh.
- ...

Other `key`/`value` pairs arguments can be used depending of `obj.d` and `obj.m` values and they are those of the plotting function used:

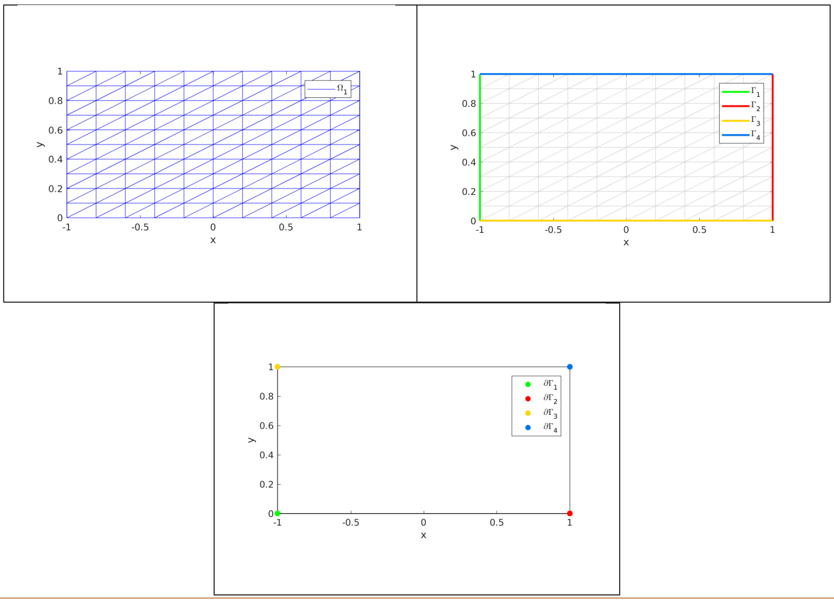
- with `obj.d=3` and `obj.m=3` , `patch` function is used;
- with `obj.d=3` and `obj.m=2` , `trimesh` function is used for simplicial mesh and `patch` function is used for orthotope mesh;
- with `obj.d=3` and `obj.m=1` , `line` function is used;
- with `obj.d=3` and `obj.m=0` , `scatter3` function is used;
- with `obj.d=2` and `obj.m=2` , `triplot` function is used for simplicial mesh and `patch` function is used for orthotope mesh;
- with `obj.d=2` and `obj.m=1` , `line` function is used;
- with `obj.d=2` and `obj.m=0` , `scatter` function is used;
- with `obj.d=1` and `obj.m=1` , `line` function is used;
- with `obj.d=1` and `obj.m=0` , `scatter` function is used;

3.2 2d-orthotope meshing by simplices

In Listing 1, an `OrthMesh` object is built under Matlab by using command

```
Oh=OrthMesh(2,10,'box',[-1,1;0,1])
```

So the `Oh` object is the tessellations of the orthotope $[-1,1]\times[0,1]$ with simplicial elements. In each direction $10+1(=11!)$ points are taken. So we have 11^2 vertices in this mesh. The main mesh and all the `m`-face meshes of the resulting object are plotted by using `plotmesh` method.

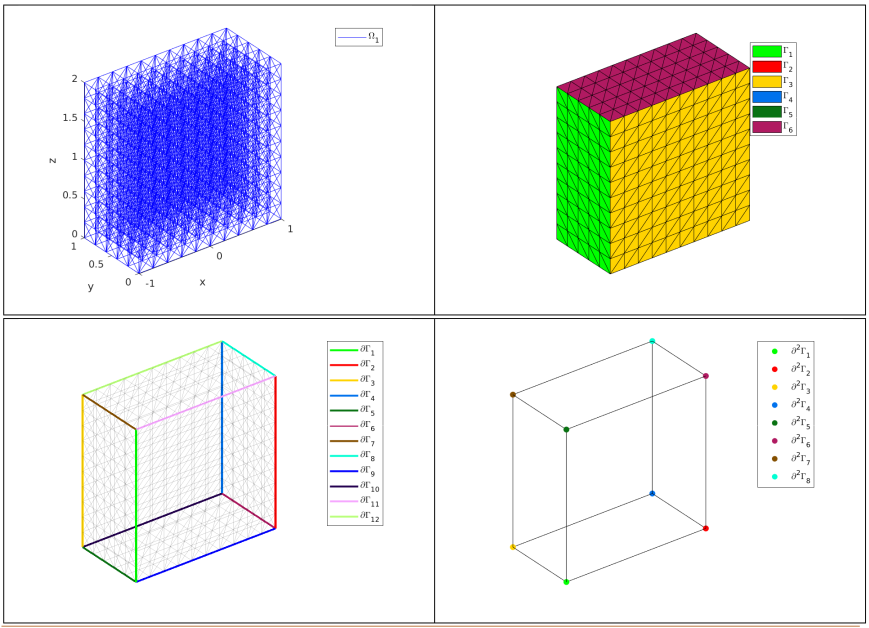



```
Oh=OrthMesh(2,10,'box',[-1,1;0,1])
% plot the main mesh
figure(1)
Oh.plotmesh('legend',true)
% plot the 1-face meshes
figure(2)
Oh.plotmesh('color',[0.8,0.8,0.8])
hold on
Oh.plotmesh('m',1,'Linewidth',2,'legend',true)
% plot the 0-face meshes
figure(3)
Oh.plotmesh('m',1,'color','k')
hold on
Oh.plotmesh('m',0,'legend',true)
```

Listing 1: 2D simplicial `OrthMesh` object with Matlab 2017a, main mesh (upper left), 1-face meshes (upper right), and 0-face meshes (bottom)

3.3 3d-orthotope meshing by simplices

In Listing 1, an `OrthMesh` object is built under Matlab for the orthotope $[-1,1]\times[0,1]\times[0,2]$ with simplicial elements and `N=[10,5,10]`. The main mesh and all the `m`-face meshes of the resulting object are plotted.

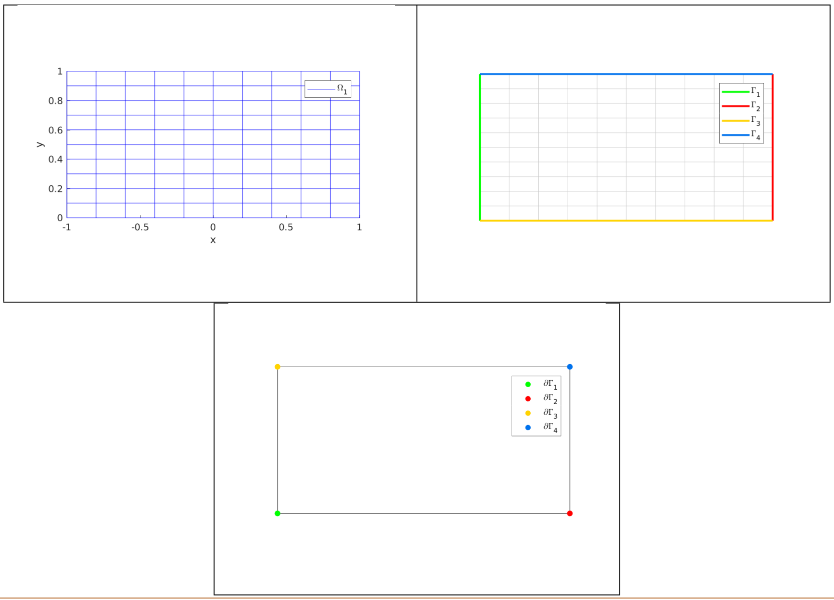



```
Oh=OrthMesh(3,[10,5,10],'box',[-1,1;0,1;0,2])
% plot the main mesh
figure(1)
Oh.plotmesh('legend',true)
axis equal;xlabel('x');ylabel('y');zlabel('z')
% plot the 2-face meshes
figure(2)
Oh.plotmesh('m',2,'legend',true)
axis equal;axis off
% plot the 1-face meshes
figure(3)
Oh.plotmesh('m',2,'color',[0.8,0.8,0.8],'EdgeAlpha',0.2, ...
    'FaceColor','none')
hold on
Oh.plotmesh('m',1,'Linewidth',2,'legend',true)
axis equal;axis off
% plot the 0-face meshes
figure(4)
Oh.plotmesh('m',1,'color','k')
hold on
Oh.plotmesh('m',0,'legend',true)
axis equal;axis off
```

Listing 2: 3D simplicial `OrthMesh` object with Matlab 2017a, main mesh (upper left), 2-face meshes (upper right), 1-face meshes (bottom left) and 0-face meshes (bottom right)

3.4 2d-orthotope meshing by orthotopes

In Listing 1, an `OrthMesh` object is built under Matlabfor the orthotope $[-1,1]\times[0,1]\times[0,2]$ with orthotope elements and `N=[10,5,10]`. The main mesh and all the `m`-face meshes of the resulting object are plotted.

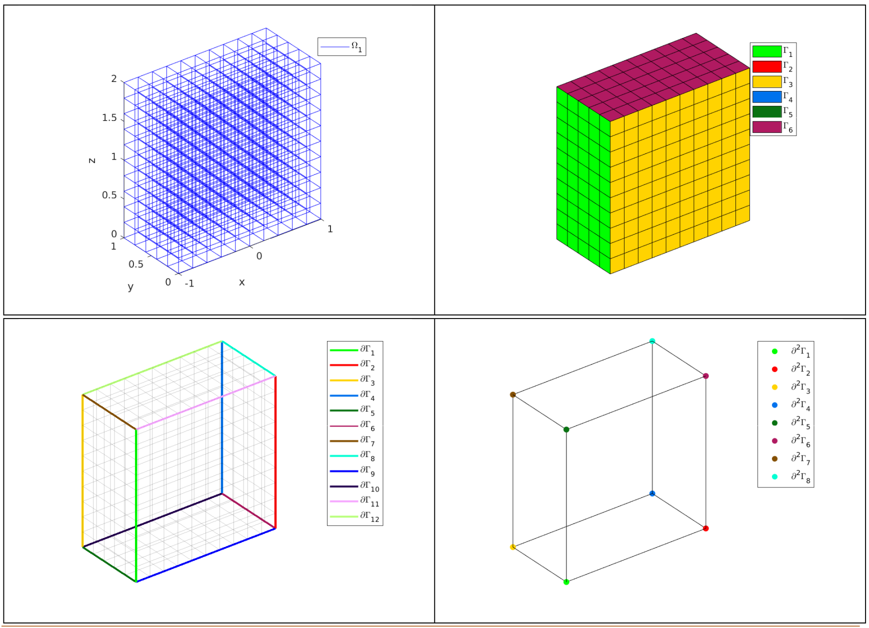



```
Oh=OrthMesh(2,10,'box',[-1,1;0,1],'type','orthotope')
% plot the main mesh
figure(1)
Oh.plotmesh('legend',true)
axis equal;xlabel('x');ylabel('y')
% plot the 1-face meshes
figure(2)
Oh.plotmesh('color',[0.8,0.8,0.8])
hold on
Oh.plotmesh('m',1,'Linewidth',2,'legend',true)
axis equal;axis off
% plot the 0-face meshes
figure(3)
Oh.plotmesh('m',1,'color','k')
hold on
Oh.plotmesh('m',0,'legend',true)
axis equal;axis off
```

Listing 3: 2D orthotope `OrthMesh` object with Matlab 2017a, main mesh (upper left), 1-face meshes (upper right), and 0-face meshes (bottom)

3.5 3d-orthotope meshing by orthotopes

In Listing 1, an `OrthMesh` object is built under Matlab for the orthotope $[-1,1]\times[0,1]\times[0,2]$ with orthotope elements and `N=[10,5,10]`. The main mesh and all the `m`-face meshes of the resulting object are plotted.



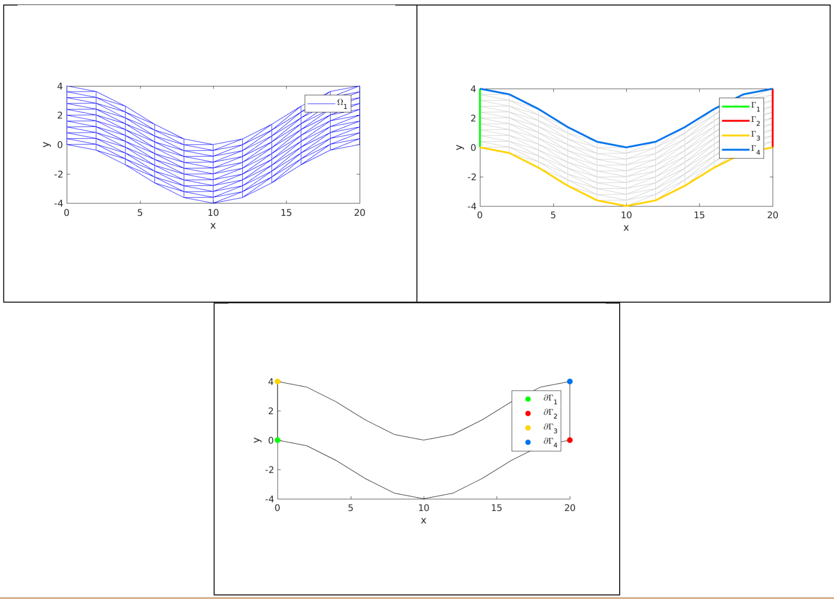

```
Oh=OrthMesh(3,[10,5,10],'box',[-1,1;0,1;0,2], 'type','orthotope')
% plot the main mesh
figure(1)
Oh.plotmesh('legend',true)
axis equal;xlabel('x');ylabel('y');zlabel('z')
% plot the 2-face meshes
figure(2)
Oh.plotmesh('m',2,'legend',true)
axis equal;axis off
% plot the 1-face meshes
figure(3)
Oh.plotmesh('m',2,'color',[0.8,0.8,0.8],'EdgeAlpha',0.2, ...
    'FaceColor','none')
hold on
Oh.plotmesh('m',1,'Linewidth',2,'legend',true)
axis equal;axis off
% plot the 0-face meshes
figure(4)
Oh.plotmesh('m',1,'color','k')
hold on
Oh.plotmesh('m',0,'legend',true)
axis equal;axis off
```

Listing 4: 3D orthotope `OrthMesh` object with Matlab 2017a, main mesh (upper left), 2-face meshes (upper right), 1-face meshes (bottom left) and 0-face meshes (bottom right)

3.6 Mapping of a 2d-orthotope meshing by simplices

For example, the following 2D geometrical transformation allows to deform the reference unit hypercube.

$$\begin{array}{rcl} [0,1]\times[0,1] & \longrightarrow & \mathbb{R}^2 \\ \begin{pmatrix} x \\ y \end{pmatrix} & \longrightarrow & F(x,y) = \begin{pmatrix} 20\,x \\ 2\,(2\,y - 1 + \cos(2\pi x)) \end{pmatrix} \end{array}$$



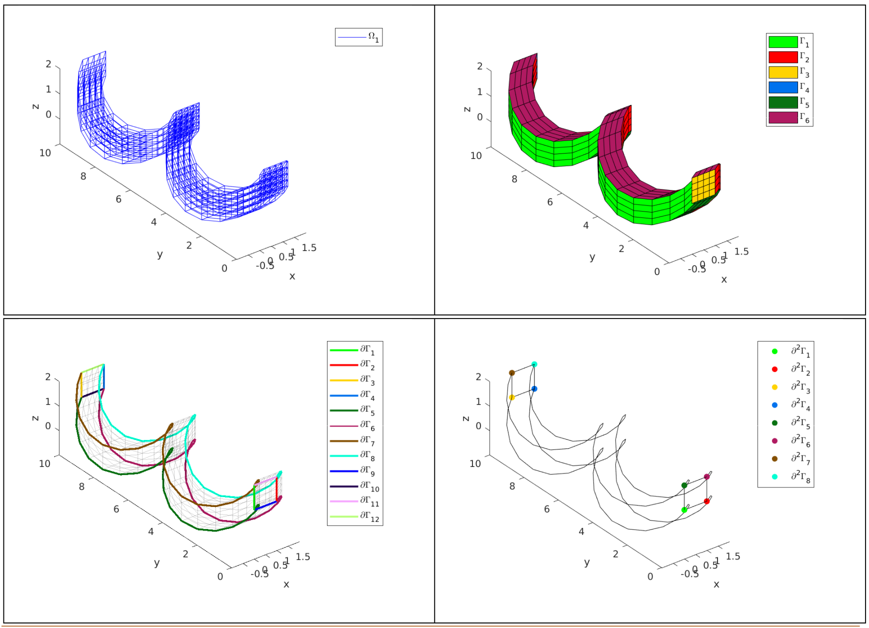

```
Oh=OrthMesh(2,10,'mapping',@(q) [20*q(1,:); ...
    2*(2*q(2,:)-1+cos(2*pi*q(1,:)))]);
% plot the main mesh
figure(1)
Oh.plotmesh('legend',true)
% plot the 1-face meshes
figure(2)
Oh.plotmesh('color',[0.8,0.8,0.8])
hold on
Oh.plotmesh('m',1,'Linewidth',2,'legend',true)
% plot the 0-face meshes
figure(3)
Oh.plotmesh('m',1,'color','k')
hold on
Oh.plotmesh('m',0,'legend',true)
```

Listing 5: Mapping of a 2D simplicial OrthMesh object with Matlab 2017a, main mesh (upper left), 1-face meshes (upper right), and 0-face meshes (bottom)

3.7 3d-orthotope meshing by orthotopes

For example, the following 3D geometrical transformation allows to deform the reference unit hypercube.

$$[0,1]\times[0,1]\times[0,1] \longrightarrow \mathbb{R}^2$$
$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \longrightarrow F(x,y,y) = \begin{pmatrix} x+\sin(4\pi y) \\ 10y \\ z+\cos(4\pi y) \end{pmatrix}$$



```
Map=@(q) [q(1,:)+ sin(4*pi*q(2,:)); 10*q(2,:); q(3,:)+cos(4*pi*q(2,:))];
Oh=OrthMesh(3,[4,25,4],'mapping',Map,'type','orthotope');
% plot the main mesh
figure(1)
Oh.plotmesh()
legend('show')
% plot the 2-face meshes
figure(2)
Oh.plotmesh('m',2)
legend('show')
% plot the 1-face meshes
figure(3)
Oh.plotmesh('m',2,'color',[0.8,0.8,0.8],'EdgeAlpha',0.2, ...
    'FaceColor','none')
hold on
Oh.plotmesh('m',1,'Linewidth',2,'legend',true)
% plot the 0-face meshes
figure(4)
Oh.plotmesh('m',1,'color','k')
hold on
Oh.plotmesh('m',0,'legend',true)
```

Listing 6: Mapping of a 3D orthotope `OrthMesh` object with Matlab 2017a, main mesh (upper left), 2-face meshes (upper right), 1-face meshes (bottom left) and 0-face meshes (bottom right)

4 Benchmarking

4.1 `fc_bench.bench01` function

The `fc_bench.bench01` function can be used to obtain computationnal times of the `OrthMesh` constructor.

Syntaxe

```
fc_bench.bench01(d,ctype,Box,LN)
```

Description

`fc_bench.bench01(d,ctype,Box,LN)`

displays computationnal times of the `OrthMesh` constructor as follows

```
ts=tic();Oh=OrthMesh(d,N,'box',Box,'type',ctype);tcpu=toc(ts);
```

for each `N` in `LN`.

4.2 Examples

Listing 7: : Computationnal times of `OrthMesh` constructor in dimension `d=3` (simplicial mesh)

```
fc_hypermesh.bench01(3,'simplicial',[-1 1;-1 1;-1 1],25:25:175)
```

Output

```
# BENCH in dimension 3 with simplicial mesh
#d: 3
#type: simplicial
#box: [ -1 1; -1 1; -1 1]
#desc: N        nq      nme    time(s)
      25    17576    93750    0.391
      50   132651   750000    0.122
      75   438976  2531250    0.196
     100  1030301  6000000    0.378
     125  2000376 11718750    0.691
     150  3442951 20250000    1.370
     175  5451776 32156250    2.116
```

Listing 8: : Computationnal times of `OrthMesh` constructor in dimension `d=5` (orthotope mesh)

```
fc_hypermesh.bench01(5,'orthotope',[-1 1;-1 1;-1 1;-1 1;-1 ...
    1],[5:5:25,27])
```

Output

```
# BENCH in dimension 5 with orthotope mesh
#d: 5
#type: orthotope
#box: [ -1 1; -1 1; -1 1; -1 1; -1 1]
#desc: N        nq      nme    time(s)
       5      7776     3125    0.525
      10    161051   100000    0.242
      15   1048576   759375    0.365
      20   4084101  3200000    0.791
      25  11881376  9765625    1.848
      27  17210368 14348907    2.606
```

5 Mesh refinement

5.1 Non-conforming mesh refinement

In this part we propose a preliminary refinement of a regular `d`-grid. We want to generate a refinement of some cells of this regular grid with `d`-simplices and/or `d`-orthotopes: the mesh obtained is therefore non-conforming.

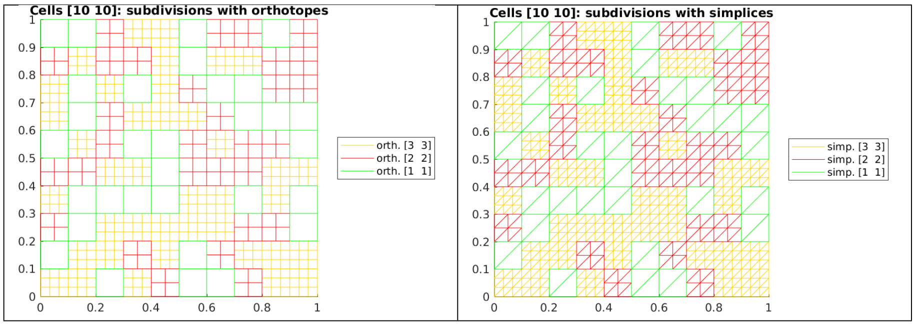


Figure 1: regular refinement of a 2D-grid with orthotopes (left) and simplices (right)

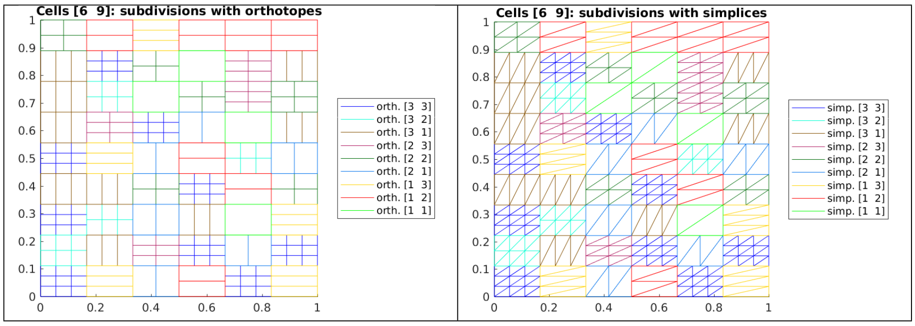


Figure 2: no regular refinement of a 2D-grid with orthotopes (left) and simplices (right)

In Figure 1, a 2D-grid with 10-by-10 cells is refined with regular subcells (i.e. same number of discretisation in each directory) made of simplices or orthotopes. For example a cell subdivided in 3-by-3 orthotopes is denoted by `orth[3 3]` and this same cell subdivided in 3-by-3 simplices (in fact $3 \times 3 \times 2$ simplices) is denoted by `simp[3 3]`. In Figure 2, a 2D-grid with 10-by-10 cells (left) and with 6-by-9 cells (right) is refined with subcells (not necessarily with same number of discretisation in each directory). In Figure 3, a 2D-grid with 10-by-10 cells (left) is refined with regular subcells made of orthotopes or simplices. On the right, a 2D-grid with 6-by-9 cells is is refined with not necessarily regular subcells made of orthotopes or simplices refined with subcells respectively.

5.1.1 Refinement function

The refinement of an `OrthMesh` object whose elements are orthotopes is obtained by using the `fc_hypermesh.refinement.refine` Matlabfunction.

```
scs = refine(Oh,ndiscells)
scs = refine(Oh,ndiscells,type)
scs = refine(Oh,ndiscells,types)
```

This function returns cells array of structures. Each array entry contains all cells

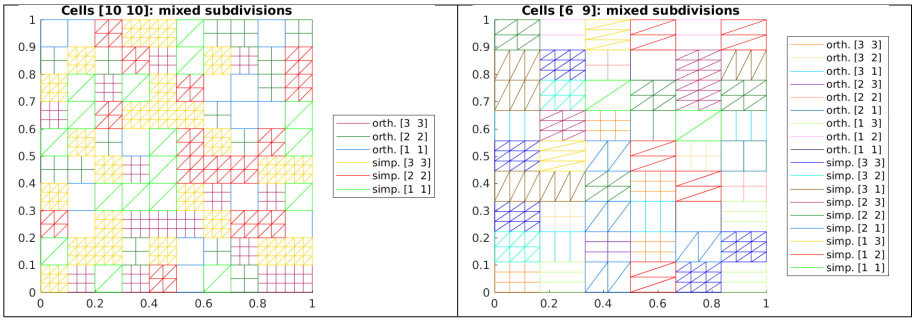


Figure 3: Regular (left) and not regular (right) refinement of a 2D-grid with orthotopes and simplices

refined with a `type` of element (simplicial or orthotope) and selected numbers of discretisation in each axis direction. Each structure (called *subcells structure*) has the fields

- `d` : space dimension
- `type` : `type` of element: `'simplicial'` or `'orthotope'`
- `n` : 1-by- `d` array corresponding to selected numbers of discretisation in each axis direction
- `nq` : number of vertices
- `nme` : number of mesh elements
- `q` : `d` -by- `nq` vertices array
- `me` : p-by- `nme` connectivity array ($p = \mathtt{d} + 1$ for simplices and $p = 2^{\mathtt{d}}$ for orthotopes)
- `ncell` : contains all the number of the cells grid which are refined.

Description

```
scs = fc_hypermesh.refinement.refine(Oh,ndiscells)
```

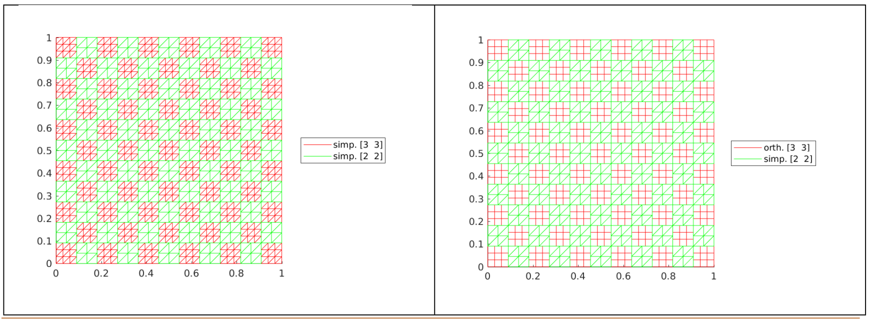
The first input `Oh` is an `OrthMesh` object whose elements are orthotopes. `ndiscells` is an 1-by-`Oh.Mesh.nme` array (for regular refinement) or a `d`-by-`Oh.Mesh.nme` array. `ndiscells(:,k)` are the numbers of discretisation in each axis direction. The output is a cells array of *subcells structure* where a all subcells `type` are `'orthotope'`

```
scs = fc_hypermesh.refinement.refine(Oh,ndiscells,type)
```

Same as previous one if `type` is `'orthotope'` . Otherwise `type` is `'simplicial'` and all subcells `type` of the output are `'simplicial'`

```
[scsimp,scorth] = fc_hypermesh.refinement.refine(Oh,ndiscells,types)
```

The input `types` is an 1-by-`Oh.Mesh.nme` array: if `types(k)` is equal to 0 then the k-th mesh element (cell) of the `OrthMesh` is refined with simplices otherwise with orthotopes.




```
d=2; % space dimension
N=11; % or N=[10,10]
Oh=OrthMesh(d,N,'type','orthotope');
nme=Oh.Mesh.nme; % Number of Mesh elements
ndiscells=2*ones(1,nme);
ndiscells(1:2:nme)=3;
sc1=fc_hypermesh.refinement.refine(Oh,ndiscells,'simplicial');
figure(1)
fc_hypermesh.refinement.plotsubcells(sc1)
set(legend(),'Location','eastoutside');axis image

types=zeros(1,nme);ndiscells=2*ones(1,nme);
types(1:2:nme)=1;ndiscells(1:2:nme)=3;
sc2=fc_hypermesh.refinement.refine(Oh,ndiscells,types);
figure(2)
fc_hypermesh.refinement.plotsubcells(sc2)
set(legend(),'Location','eastoutside');axis image
```

Listing 9: 2D orthotope `OrthMesh` refinement with Matlab 2017a, figure 1 (left) and figure 2 (right)

More examples are provided by scripts:

- `fc_hypermesh.refinement.demo01`
- ...
- `fc_hypermesh.refinement.demo10`