hypermesh Octave package, User's Guide*

version 1.0.0

François Cuvelier†

March 6, 2019

Abstract

This object-oriented Octave package allows in any dimension d to generate conforming meshes of hypercubes, hyperrectangles or of any d-orthotopes by p-order simplices or orthotopes with their m-faces. It was created to show the implementation of the algorithms of [1]. The hypermesh package uses Octave objects and is provided with meshes visualisation tools for dimension less than or equal to 3.

0 Contents

1 Introduction **2**

2 Installation **8**

3 Installation via pkg command **8**

4 All-in-one installation **9**

*Manual revision 1.0.0.a, compiled with Octave 5.1.0, and packages `fc-hypermesh`[1.0.0], `fc-tools`[0.0.25], `fc-bench`[0.0.6]

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

5 Using the `fc_hypermesh` package **10**
5.1 Class object OrthMesh 11
5.2 2d-orthotope meshed by simplices 22
5.3 3d-orthotope meshed by simplices 23
5.4 2d-orthotope meshed by orthotopes 24
5.5 3d-orthotope meshed by orthotopes 25
5.6 Mapping of a 2d-orthotope meshed by simplices 26
5.7 3d-orthotope meshed by orthotopes 27

6 Benchmarking **28**
6.1 fc_hypermesh.bench function 28
6.2 Examples 29

7 Mesh refinement **31**
7.1 Non-conforming mesh refinement 31

Appendices **34**

A Memory consuming **34**

B Low level codes **35**
B.1 Class object EltMesh 35

1 Introduction

The fc hypermesh package contains a simple class object OrthMesh which permits, in any dimension $\mathrm{d} \geqslant 1$, to obtain conforming mesh of a d-orthotope tessellated with p-order simplices or p-order orthotopes. Corresponding m-faces, $0 \leqslant \mathrm{m} < \mathrm{d}$ of the mesh are also provided. The number of m-faces of a d-orthotope is

$$E_m^{\mathrm{d}} \stackrel{\text{def}}{=} 2^{\mathrm{d}-m} \binom{d}{m} \quad \text{where} \quad \binom{\mathrm{d}}{m} = \frac{\mathrm{d}!}{m!(\mathrm{d}-m)!} \tag{1}$$

Results and vectorized algorithms used in this package are given in [1].

For dimension 1 to 3 and order 1 to 4, orthotope elements and simplicial elements are respectively represented In Table 1 and Table 2. In older package(0.0.x versions) only order 1 was provided.

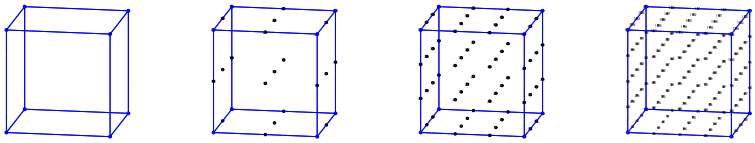
d \ p	1	2	3	4
1				
2				
3				

Table 1: p-order d-orthotope mesh element in $\mathbb{R}^d$. Nodes are the points.

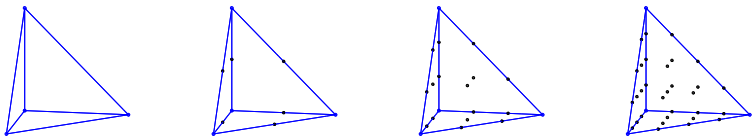
d \ p	1	2	3	4
1				
2				
3				

Table 2: p-order d-simplicial mesh element in $\mathbb{R}^d$. Nodes are the points.

In Figure 1 and Figure 3 small meshes of the unit hypercube are given for both tessellations with 1-order orthotopes and 1-order simplices respectively in dimension 2 and 3. From theses meshes, all the associated 2-faces meshes are represented in Figure 2 and Figure 4.

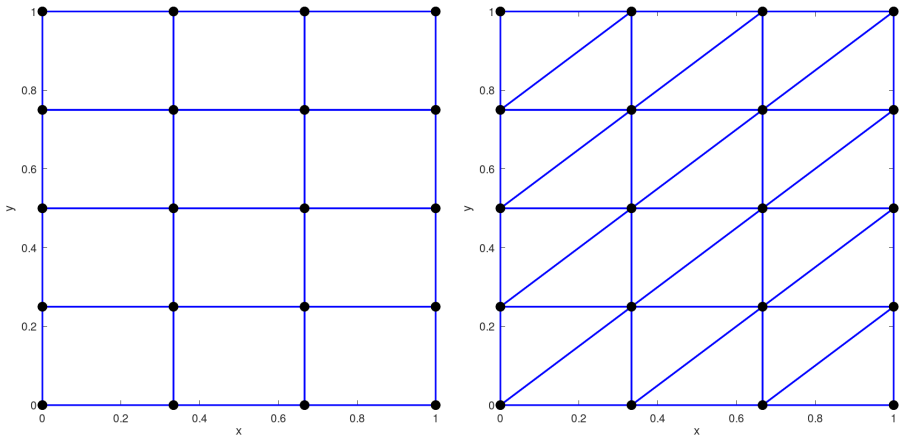


Figure 1: Tesselation samples of $[0,1]^2$ with 1-order 2-orthotopes (left) and 1-order 2-simplices (right) where nodes (vertices) of all mesh elements are represented by black points.

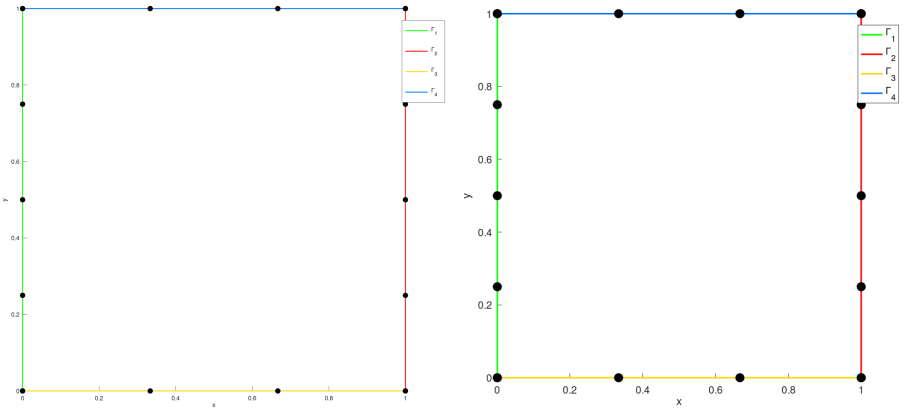


Figure 2: Representation of all the 1-faces meshes with 1-order 1-orthotopes (left) and 1-order 1-simplices (right) obtained from the tesselation samples of the Figure 1

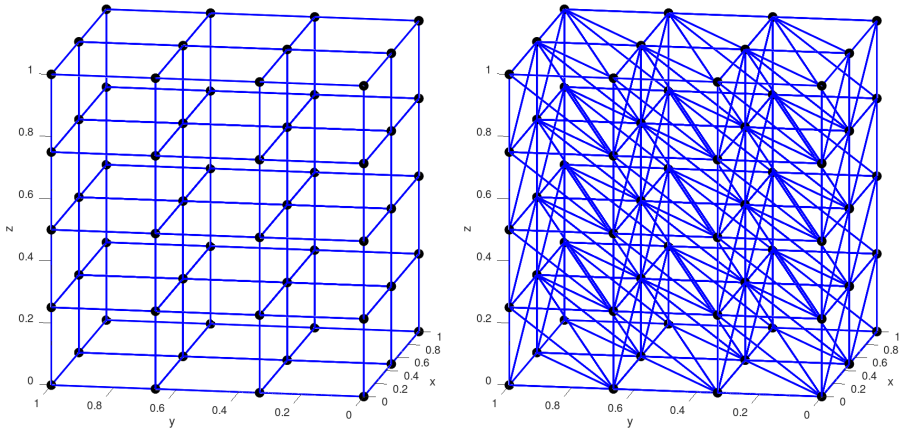


Figure 3: Tesselation samples of $[0,1]^3$ with 1-order 3-orthotopes (left) and 1-order 3-simplices (right) where nodes (vertices) of all mesh elements are represented by black spheres.

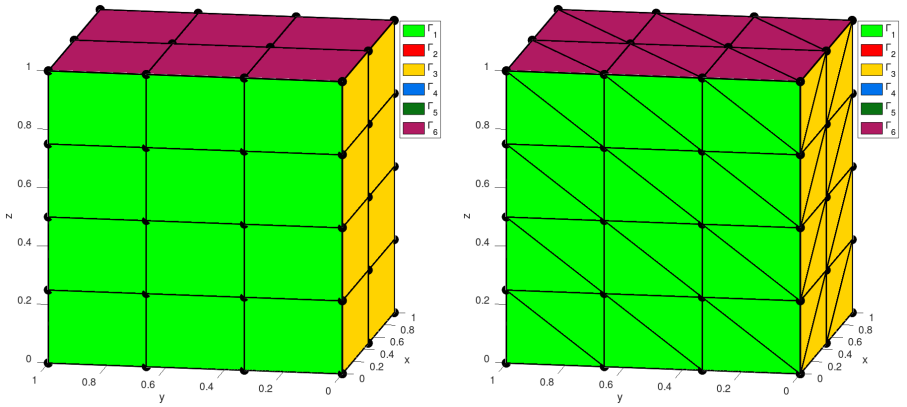


Figure 4: Representation of all the 2-faces meshes with 1-order 2-orthotopes (left) and 1-order 2-simplices (right) obtained from the tesselation samples of the Figure 3

By taking back the meshes in dimension 2 represented in Figure 1 and Figure 2, but this time using 3-order mesh element give the new meshes represented in Figure 5 and Figure 6.

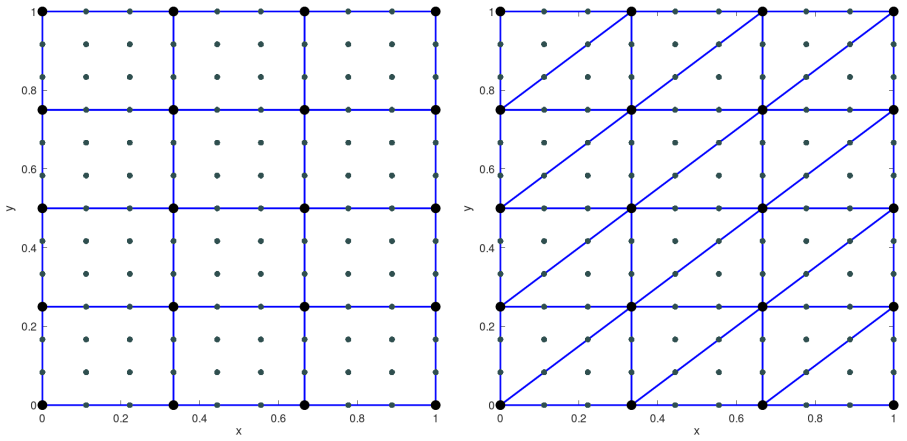


Figure 5: Tesselation samples of $[0,1]^2$ with 3-order 2-orthotopes (left) and 3-order 2-simplices (right) where nodes of all mesh elements are represented by black (vertices) and grey points.

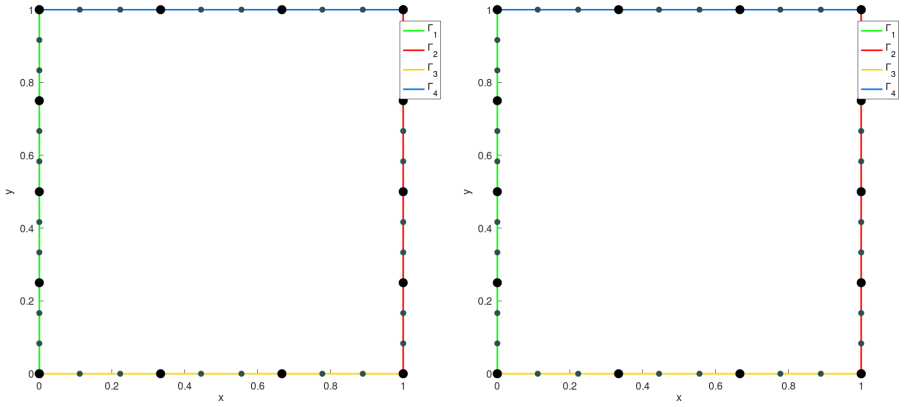


Figure 6: Representation of all the 1-faces meshes with 3-order 1-orthotopes (left) and 3-order 1-simplices (right) obtained from the tesselation samples of the Figure 5

In dimension 3, meshes represented in Figure 3 and Figure 4 are this time tessellated respectively with 3-order orthotopes and 3-order simplices and represented in Figure 7 and Figure 8.

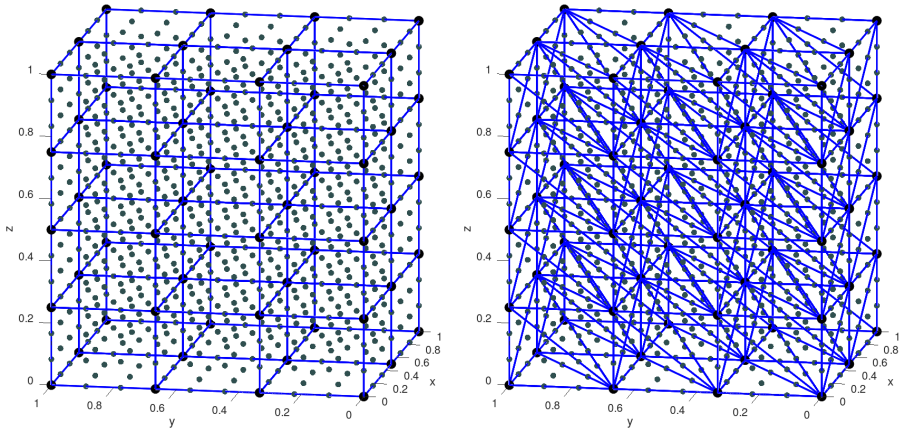


Figure 7: Tesselation samples of $[0,1]^3$ with 3-order 3-orthotopes (left) and 3-order 3-simplices (right) where nodes of all mesh elements are represented by black (vertices) and grey spheres.

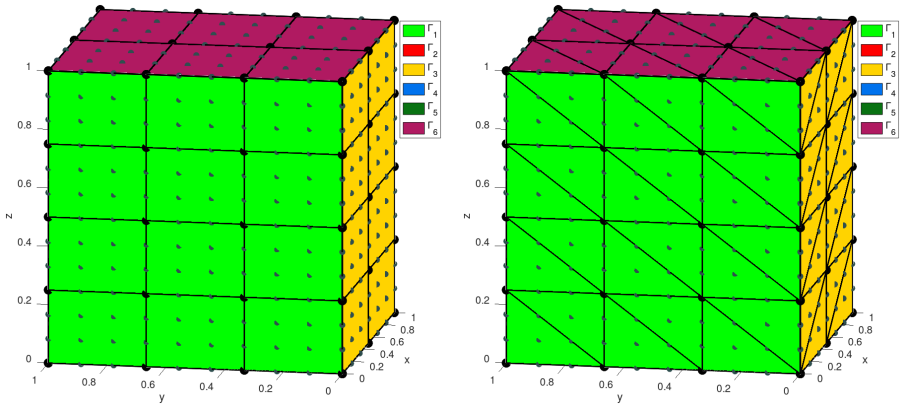


Figure 8: Representation of all the 2-faces meshes with 3-order 2-orthotopes (left) and 3-order 2-simplices (right) obtained from the tesselation samples of the Figure 7

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 package was tested on

Windows 10 (1803): with Octave versions 4.2.x ($x = 0, 1, 2$), 4.4.x ($x = 0, 1$) and 5.1.0

macOS High Sierra 10.13.6: with Octave 4.2.2 (installed with homebrew) 4.4.x ($x = 0, 1$) (installed from `http://octave-app.org/`)

macOS Mojave 10.14: with Octave 4.4.x ($x = 0, 1$) (installed from `http://octave-app.org/`)

Ubuntu 18.04.2 LTS: with Octave versions 4.2.x ($x = 0, 1, 2$), 4.4.x ($x = 0, 1$) and 5.1.0

centOS 7.6.1810: with Octave versions 4.2.x ($x = 0, 1, 2$), 4.4.x ($x = 0, 1$) and 5.1.0

Fedora 29: with Octave versions 4.2.x ($x = 0, 1, 2$), 4.4.x ($x = 0, 1$) and 5.1.0

OpenSUSE Leap 15.0: with Octave versions 4.2.x ($x = 0, 1, 2$), 4.4.x ($x = 0, 1$) and 5.1.0

Debian 9.8: with Octave versions 4.2.x ($x = 0, 1, 2$), 4.4.x ($x = 0, 1$) and 5.1.0

It is not compatible with Octave 4.0.x and previous.

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 $\mathrm{d} \in [\![1, 5]\!]$.

2 Installation

Here are two methods of installations. The first uses the Octave `pkg` command and the second a provided Octave script.

3 Installation via pkg command

- Download the packages. For example, in a terminal:

```
wget http://www.math.univ-paris13.fr/~cuvelier/software/codes/
    Octave/fc-tools/0.0.25/fc-tools-0.0.25.tar.gz
wget http://www.math.univ-paris13.fr/~cuvelier/software/codes/
    Octave/fc-bench/0.0.6/fc-bench-0.0.6.tar.gz
wget http://www.math.univ-paris13.fr/~cuvelier/software/codes/
    Octave/fc-hypermesh/1.0.0/fc-hypermesh-1.0.0.tar.gz
```

- Under Octave :

```
>> pkg install fc-tools-0.0.25.tar.gz
>> pkg install fc-bench-0.0.6.tar.gz
>> pkg install fc-hypermesh-1.0.0.tar.gz
```

  With all Octave releases beginning with `4.4.0`, one can directly use

```
pkg install http://www.math.univ-paris13.fr/~cuvelier/software/
    codes/Octave/fc-tools/0.0.25/fc-tools-0.0.25.tar.gz
pkg install http://www.math.univ-paris13.fr/~cuvelier/software/
    codes/Octave/fc-bench/0.0.6/fc-bench-0.0.6.tar.gz
pkg install http://www.math.univ-paris13.fr/~cuvelier/software/
    codes/Octave/fc-hypermesh/1.0.0/fc-hypermesh-1.0.0.tar.gz
```

- Now to use `fc-hypermesh` in any Octave session, it is necessary to load and initialize the package :

```
>> pkg load fc-hypermesh
>> fc_hypermesh.init()
```

- To try the package, one can launch a demo:

```
>> fc_hypermesh.demos.demo01
```

For uninstalling the package, just do in an Octave session:

```
>> pkg uninstall fc-hypermesh
>> pkg uninstall fc-bench
>> pkg uninstall fc-tools
```

4 All-in-one installation

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

ofc_hypermesh_install.m

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

For example, to install this package in directory `~/Octave/packages`, in a terminal one can do:

```
# mkdir -p ~/Octave/packages
# cd ~/Octave/packages
# wget http://www.math.univ-paris13.fr/~cuvelier/software/codes/Octave/fc
    -hypermesh/1.0.0/ofc_hypermesh_install.m
```

Then in a Octave terminal run the following commands

```
>> cd ~/Octave/packages
>> ofc_hypermesh_install
```

This is the output of the `ofc_hypermesh_install` command:

```
Parts of the <fc-hypermesh> Octave package.
Copyright (C) 2016-2019 F. Cuvelier

1- Downloading and extracting the packages
2- Setting the <fc-hypermesh> package
Write in ~/Octave/packages/fc-hypermesh-full/fc_hypermesh-1.0.0/
    configure_loc.m ...
3- Using packages :
   ->             fc-tools : 0.0.25
   ->             fc-bench : 0.0.6
*** Using instructions
   To use the <fc-hypermesh> package:
   addpath('~/Octave/packages/fc-hypermesh-full/fc_hypermesh-1.0.0')
   fc_hypermesh.init()

   See ~/Octave/packages/ofc_hypermesh_set.m
```

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

```
>> addpath('~/Octave/packages/fc-hypermesh-full/ofc-hypermesh-1.0.0')
>> fc_hypermesh.init()
```

To **uninstall**, one just has to delete directory `~/Octave/packages/fc-hypermesh-full`

5 Using the fc hypermesh package

Before using this class it will be necessary to be aware of the memory used by this one. For example, when meshing a **??**-dimensional orthotope with 1-order simplices by taking `N` = **??** intervals in each space direction, gives an `OrthMesh` object using **??** in memory. With **??**-order simplices, the `OrthMesh` object use **??** in memory!

The memory usage for a `d`-dimensional `OrthMesh` object by taking `N` = **??** intervals in each space direction is given in Table 3 for 1-order elements and in Table 4 for **??**-order elements. One can refer to Appendix A for more details.

`d`	`OrthMesh` (orthotopes)	`OrthMesh` (simplices)
1	??	??
2	??	??
3	??	??
4	??	??
5	??	??
6	??	??
7	??	??
8	??	??

Table 3: Memory usage of `OrthMesh` object for the tessellation of an orthotope by 1-order orthotopes and by 1-order simplices according to the space dimension `d` and with N = **??**.

`d`	`OrthMesh` (orthotopes)	`OrthMesh` (simplices)
1	??	??
2	??	??
3	??	??
4	??	??
5	??	??
6	??	??
7	??	??
8	??	??

Table 4: Memory usage of `OrthMesh` object for the tessellation of an orthotope by **??**-order orthotopes and by **??**-order simplices according to the space dimension `d` and with N = **??**.

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

5.1 Class object `OrthMesh`

The aim of the `OrthMesh` class object is to efficiently create an object which contains a mesh of a `d`-orthotope and all its `m`-face meshes with either `p`-order orthotopes or `p`-order simplices. An elementary mesh class object called `EltMesh` is used to store only one mesh, the main mesh as well as any of the `m`-face meshes. The `EltMesh` class is described in Appendix B.1.

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 `'simplex'` or `'orthotope'` mesh
- `p`, order of the mesh elements
- `Mesh`: main mesh as an `EltMesh` object
- `Faces`: cell array of arrays of `EltMesh` objects such that `Faces{1}` is an array of all the $(\mathtt{d}-1)$-face meshes stored as `EltMesh` objects, `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)` $= b_i$.

5.1.1 Constructor

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

Description

`Oh = fc_hypermesh.OrthMesh(N)`

Genrerates the `OrthMesh` object `Oh` which contains a 1-order simplicial mesh of the unit `d`-orthotope and all its `m`-face meshes. `N` is a 1-by-`d` array used to obtain a regular discretisation with `N`$(i)+1$ points in the i-th direction.

`Oh = fc_hypermesh.OrthMesh(d,N)`

Genrerates the `OrthMesh` object `Oh` which contains a 1-order simplicial mesh of the unit `d`-orthotope and all its `m`-face meshes.

- `d` is the space dimension
- `N` is a 1-by-`d` array used to obtain a regular discretisation with `N`$(i)+1$ points in the i-th direction. If `N` is a positive integer then `N`$+1$ points are created in each direction.

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

```
Oh = fc_hypermesh.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 'simplex' and otherwise 'orthotope' can be used.

Listing 1: : OrthMesh constructor in dimension d=3 (orthotope mesh)

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

Output

```
Oh =

Mesh type : orthotope
dimension : 3
order     : 1
nq        : 1331
nme       : 1000
Number of 2-faces : 6
Number of 1-faces : 12
Number of 0-faces : 8
```

- 'order' : gives the order of the mesh elements (default is 1).

Listing 2: : OrthMesh constructor in dimension d=3 (simplicial mesh)

```
Oh = fc_hypermesh.OrthMesh(3,[10,20,15], 'order',4)
```

Output

```
Oh =

Mesh type : simplex
dimension : 3
order     : 4
nq        : 202581
nme       : 18000
Number of 2-faces : 6
Number of 1-faces : 12
Number of 0-faces : 8
```

- '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).

Listing 3: : OrthMesh constructor in dimension d=3 (simplicial mesh)

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

Output

```
Oh =

Mesh type : simplex
dimension : 3
order     : 1
nq        : 1331
nme       : 6000
Number of 2-faces : 6
Number of 1-faces : 12
Number of 0-faces : 8
```

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

Listing 4: : `OrthMesh` constructor in dimension `d=3` (simplicial mesh)

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

Output

```
Oh =

Mesh type : simplex
dimension : 3
order     : 1
nq        : 1331
nme       : 6000
Number of 2-faces : 6
```

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

Listing 5: : `OrthMesh` constructor in dimension `d=3` (simplicial mesh)

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

Output

```
Oh =

Mesh type : simplex
dimension : 3
order     : 1
nq        : 1331
nme       : 6000
Number of 2-faces : 6
Number of 1-faces : 12
Number of 0-faces : 8
```

5.1.2 Access to `OrthMesh`'s fields

In all examples given in this section, `Oh` is the `OrthMesh` object given by

```
Oh = fc_hypermesh.OrthMesh(3,10,'order',2)
```

Output

```
Oh =

Mesh type : simplex
dimension : 3
order     : 2
nq        : 9261
nme       : 6000
Number of 2-faces : 6
Number of 1-faces : 12
Number of 0-faces : 8
```

It's a 3 dimensional mesh of the unit cube tesselated with 2-order simplices where their vertices are in $(i/10, j/10, k/10)$ for all $(i, j, k) \in [\![0, 10]\!]$. The main mesh given as an `EltMesh` object is `Oh.Mesh`

```
Th=Oh.Mesh
```

Output

```
Th =

EltMesh object
   type : simplex
  order : 2
      d : 3
      m : 3
      q : (3,9261)
     me : (10,6000)
```

The k^{th} m-faces of `Oh` stored as an `EltMesh` object is given by `Oh.Faces{Oh.d-m}(k)`.

```
Fh=Oh.Faces{1}(3)
```

Output

```
Fh =

EltMesh object
   type : simplex
  order : 2
      d : 3
      m : 2
      q : (3,441)
     me : (6,200)
```

One can easily access to each field of an `EltMesh` object (see description in Appendix B.1). For example, to acces the nodes array and the connectivity array of the `Th` `EltMesh` object we do respectively `Th.q` and `Th.me`. We also have the following link between global nodes array `Th.q` and local nodes array `Fh.q`:

```
Th.q(:,Fh.toGlobal)==Fh.q
```

or more generaly, for all `m` in $[\![0, Oh.d]\!]$ and for all `k` in $[\![1, E_m^{\mathtt{d}}]\!]$

```
Oh.Mesh.q(:,Oh.Faces{Oh.d-m}(k).toGlobal)==Oh.Faces{Oh.d-m}(k).q
```

where $E_m^{\mathtt{d}}$ is defined in (1).

5.1.3 `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()`

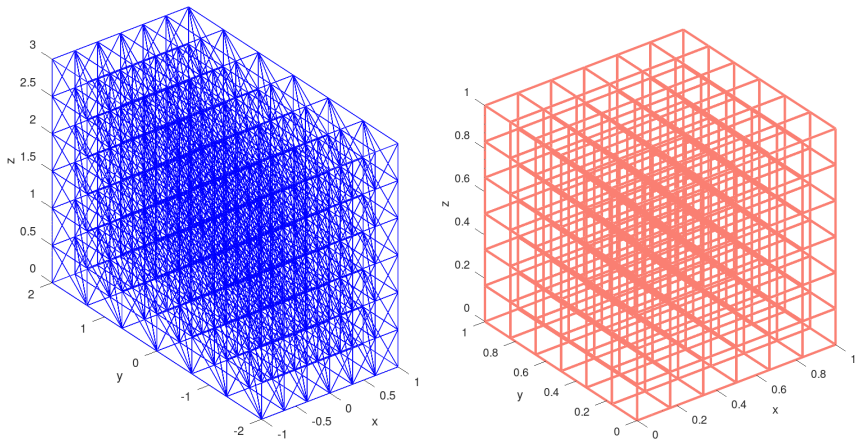
`obj.plotmesh(key, value, ...)`

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`)
- `'labels'` : plot all the `m`-faces of the mesh with number/label in `value` list
- `'color'` : use to specify the color to use.
- ...

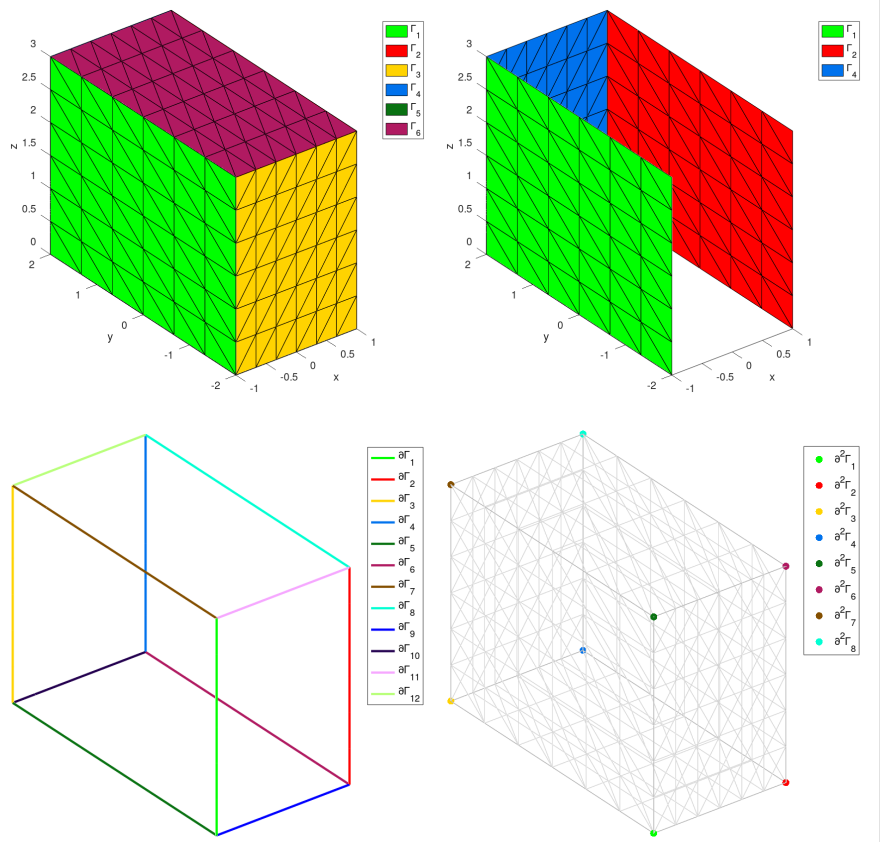
Other `key`/`value` pairs arguments can be used depending of `obj.d` and `obj.m` values and they are those of the plotting function used:

- 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;




```
Oh1=fc_hypermesh.OrthMesh(3,6,'box',[-1 1;-2 2;0 3]);
figure(1)
Oh1.plotmesh()
axis image
Oh2=fc_hypermesh.OrthMesh(3,6,'type','orthotope');
figure(2)
Oh2.plotmesh('Color','Salmon','LineWidth',2)
axis image
```

Listing 6: `plotmesh` method of 3D `OrthMesh` objects tessellated with simplices in `figure(1)` (left) and with orthotopes in `figure(2)` (right)



```
Oh=fc_hypermesh.OrthMesh(3,6,'box',[-1 1;-2 2;0 3]);
figure(1)
Oh.plotmesh('m',2,'legend',true)
axis image
figure(2)
Oh.plotmesh('m',2,'legend',true,'labels',[1,2,4])
axis image
figure(3)
Oh.plotmesh('m',1,'legend',true,'LineWidth',2)
axis image;axis off
figure(4)
Oh.plotmesh('m',2,'FaceColor','none','EdgeColor','LightGrey')
hold on
Oh.plotmesh('m',1,'color','DarkGrey')
Oh.plotmesh('m',0,'legend',true,'LineWidth',2)
axis image;axis off
```

Listing 7: `plotmesh` method of a 3D `OrthMesh` object, `figure(1)` (upper left), `figure(2)` (upper right), `figure(3)` (bottom left), `figure(4)` (bottom right)

5.1.4 `plotnodes` method

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

Syntaxe

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

Description

`obj.plotnodes()`

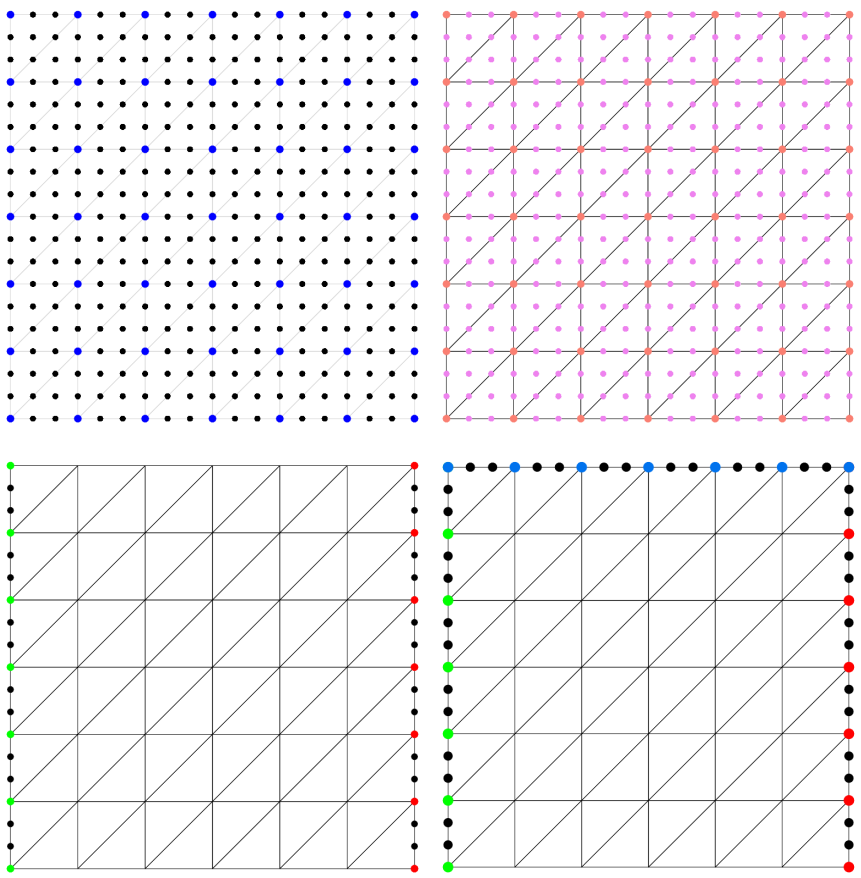
Uses `scatter` function (1D and 2D) and `scatter3` function (3D) to represent nodes of the mesh as points. Vertices of the mesh elements are also nodes and they are distinguishable from others nodes.

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

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

- `'m'` : plots all the nodes of the `m`-faces of the mesh. Default `m = d` i.e. the main mesh. ($0 \leqslant$ `m` $\leqslant$ `d`)
- `'labels'` : plot all the nodes of the `m`-faces of the mesh with number/label in `value` list
- `'vcolor'` : use to specify the point color for the mesh vertices. Default is `obj.color` .
- `'vsize'` : use to specify the point size for the mesh vertices. Default is `40` .
- `'ncolor'` : use to specify the color of the nodes (not vertices) of the mesh elements. Default is `'k'` (ie. black).
- `'nsize'` : use to specify the size of the nodes (not vertices) of the mesh elements. Default is `30` .

Other `key`/`value` pairs arguments can be used: they are those of the `scatter` and `scatter3` function used.



```
Oh=fc_hypermesh.OrthMesh(2,6,'order',3);
figure(1)
Oh.plotmesh('color','LightGray')
hold on
Oh.plotnodes()
axis image;axis off
figure(2)
Oh.plotmesh('color','k')
hold on
Oh.plotnodes('vColor','Salmon','nColor','Violet')
axis image;axis off
figure(3)
Oh.plotmesh('color','k')
hold on
Oh.plotnodes('m',1,'labels',[1,2])
axis image;axis off
figure(4)
Oh.plotmesh('color','k')
hold on
Oh.plotnodes('m',1,'labels',[1,2,4],'vSize',80,'nSize',60)
axis image;axis off
```

Listing 8: `plotnodes` method of a 2D `OrthMesh` object, `figure(1)` (upper left), `figure(2)` (upper right), `figure(3)` (bottom left), `figure(4)` (bottom right)

5.1.5 `plotnodesNumber` method

The `plotnodesNumber()` member function can be used to display nodes index/number of the mesh given by an `OrtMesh` object if the space dimension `d` is less than

or equal to 3.

Syntaxe

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

Description

`obj.plotnodesNumber()`

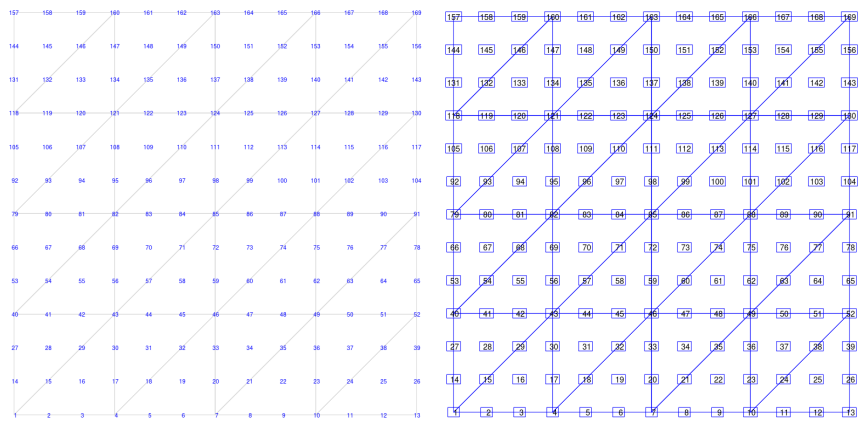
Uses `fc_hypermesh.plotnodesNumber` function to represent node numbers.

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

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

- `'m'` : plots all the nodes index/number of the `m`-faces of the mesh. Default `m = d` i.e. the main mesh. ($0 \leqslant$ `m` $\leqslant$ `d`)
- `'labels'` : plot all the nodes index/number of the `m`-faces of the mesh with number/label in `value` list
- `'color'` : use to specify text color. Default is the color element.
- `'EdgeColor'` : use to specify the color of box outline. Default is `'none'` (i.e. no box outline).
- `'BackgroundColor'` : use to specify text background color. Default is `'w'` (i.e. white).
- `'margin'` : use to specify the margin. Default is `1` .
- `'HorizontalAlignment'` : Default is `'center'` .
- `'VerticalAlignment'` : Default is `'middle'` .
- `'clipping'` : Default is `'on'` .

Other `key`/`value` pairs arguments can be used: they are those of the `text` function used. For the color options `'Color'`, `'EdgeColor'` and `'BackgroundColor'`, one can use `'element'` as `value` to set color to the element color.



```
Oh=fc_hypermesh.OrthMesh(2,4,'order',3);
figure(1)
Oh.plotmesh('color','LightGray')
hold on
Oh.plotnodesNumber()
axis image; axis off
figure(2)
Oh.plotmesh()
hold on
Oh.plotnodesNumber('Color','k','EdgeColor','element','FontSize',7)
axis image;axis off
```

Listing 9: `plotnodesNumber` method of a 2D `EltMesh` objects , `figure(1)` (left) and `figure(2)` (right)

5.1.6 `ploteltsNumber` method

The `ploteltsNumber()` member function can be used to display elements index/number of the mesh given by an `OrthMesh` object if the space dimension `d` is less than or equal to 3.

Syntaxe

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

Description

`obj.ploteltsNumber()`

Uses `fc_hypermesh.ploteElementsNumber` function to represent node numbers.

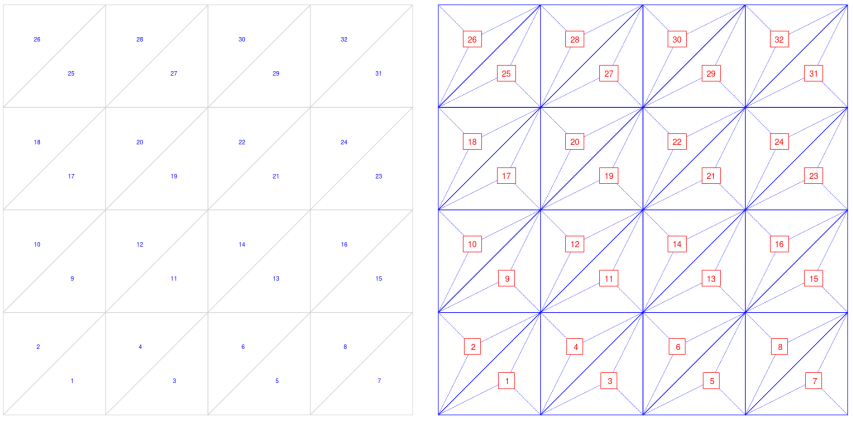
`obj.ploteltsNumber(key, value, ...)`

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

- `'m'` : plots all the elements index/number of the `m`-faces of the mesh. Default `m = d` i.e. the main mesh. ($0 \leqslant$ `m` $\leqslant$ `d`)
- `'labels'` : plot all the elements index/number of the `m`-faces of the mesh with number/label in `value` list.

- 'color' : use to specify text color. Default is the color element.
- 'EdgeColor' : use to specify the color of box outline. Default is 'none' (i.e. no box outline).
- 'BackgroundColor' : use to specify text background color. Default is 'w' (i.e. white).
- 'margin' : use to specify the margin. Default is 1 .
- 'HorizontalAlignment' : Default is 'center' .
- 'VerticalAlignment' : Default is 'middle' .
- 'clipping' : Default is 'on' .
- 'vLineColor' : Draw lines beetween vertices and barycenter of the mesh elements. Default is '' (no lines).
- 'vLineStyle' : Select lines type. Default is ':' (dotted lines).
- 'vLineWidth' : Set lines witdh. Default is 0.5 .

Other key/value pairs arguments can be used: they are those of the text function used. For the color options 'Color', 'EdgeColor', 'BackgroundColor' and 'vLineColor', one can use 'element' as value to set color to the element color.



```
Oh=fc_hypermesh.OrthMesh(2,4,'order',3);
figure(1)
Oh.plotmesh('color','LightGray')
hold on
Oh.ploteltsNumber()
axis image; axis off
figure(2)
Oh.plotmesh()
hold on
Oh.ploteltsNumber('Color','r','EdgeColor','r', 'FontSize',7, ...
    'vLineColor','element', 'BackgroundColor','w')
axis image;axis off
```

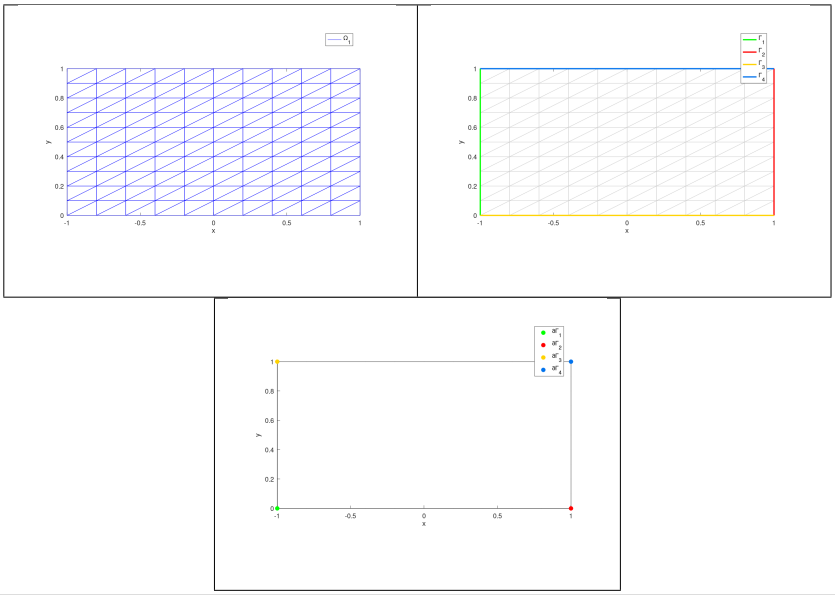
Listing 10: ploteltsNumber method of a 2D EltMesh objects , figure(1) (left) and figure(2) (right)

5.2 2d-orthotope meshed by simplices

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

```
Oh=fc_hypermesh.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.

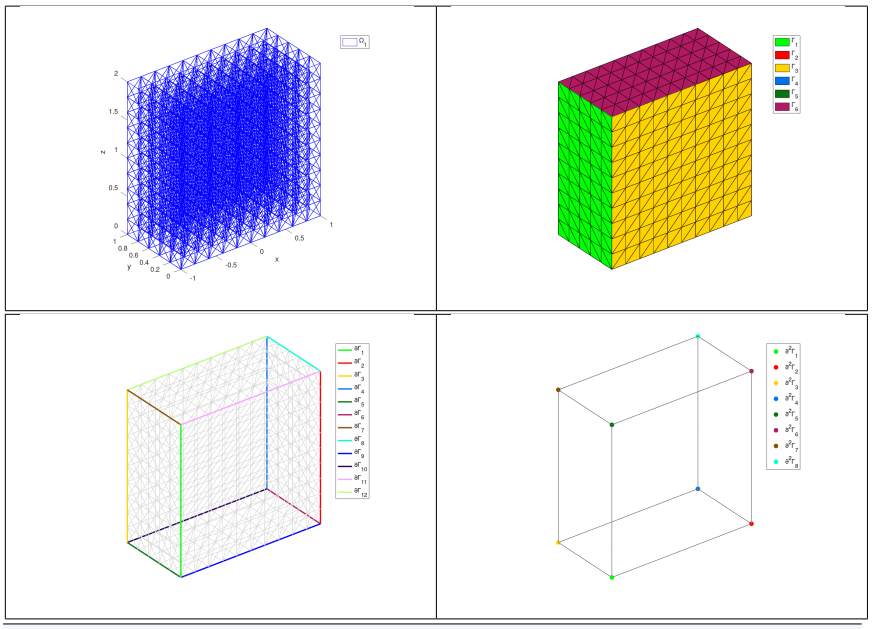


```
Oh=fc_hypermesh.OrthMesh(2,10,'box',[-1,1;0,1]);
% Drawing of the main mesh
figure(1)
Oh.plotmesh('legend',true)
% Drawing of the 1-face meshes
figure(2)
Oh.plotmesh('color',[0.8,0.8,0.8])
hold on
Oh.plotmesh('m',1,'Linewidth',2,'legend',true)
% Drawing of the 0-face meshes
figure(3)
Oh.plotmesh('m',1,'color','k')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

Listing 11: 2D simplicial `OrthMesh` object with Octave 5.1.0, main mesh (upper left), 1-face meshes (upper right), and 0-face meshes (bottom)

5.3 3d-orthotope meshed by simplices

In Listing 1, an `OrthMesh` object is built under Octave 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.

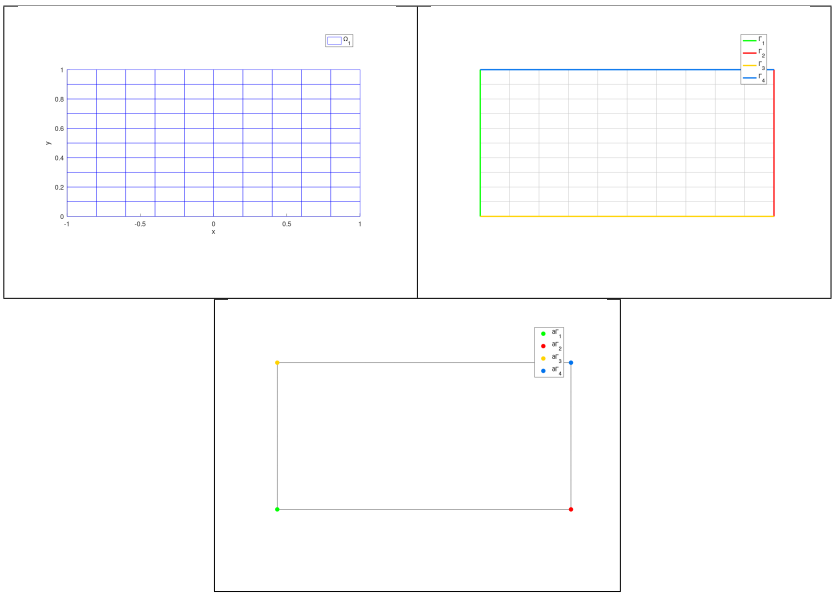


```
Oh=fc_hypermesh.OrthMesh(3,[10,5,10],'box',[-1,1;0,1;0,2]);
% plot the main mesh
figure(1)
Oh.plotmesh('legend',true)
axis equal;xlabel('x');ylabel('y');zlabel('z')
% plot the 2-face meshes
figure(2)
Oh.plotmesh('m',2,'legend',true)
axis equal;axis off
% plot the 1-face meshes
figure(3)
Oh.plotmesh('m',2,'color',[0.8,0.8,0.8],'EdgeAlpha',0.2, '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 12: 3D simplicial `OrthMesh` object with Octave 5.1.0, main mesh (upper left), 2-face meshes (upper right), 1-face meshes (bottom left) and 0-face meshes (bottom right)

5.4 2d-orthotope meshed by orthotopes

In Listing 1, an `OrthMesh` object is built under Octavefor 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.

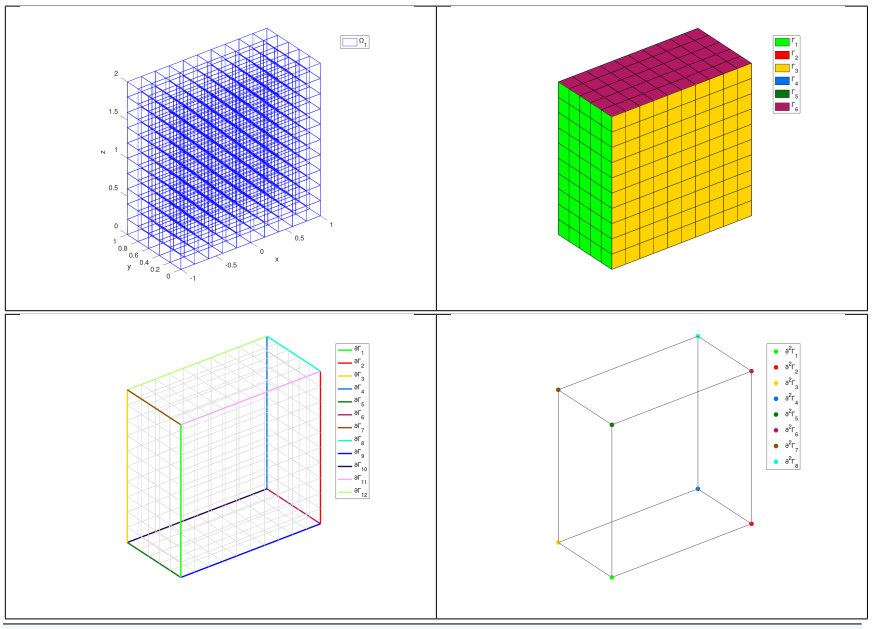


```
Oh=fc_hypermesh.OrthMesh(2,10,'box',[-1,1;0,1],'type','orthotope');
% plot the main mesh
figure(1)
Oh.plotmesh('legend',true)
axis equal;xlabel('x');ylabel('y')
% plot the 1-face meshes
figure(2)
Oh.plotmesh('color',[0.8,0.8,0.8])
hold on
Oh.plotmesh('m',1,'Linewidth',2,'legend',true)
axis equal;axis 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 13: 2D orthotope `OrthMesh` object with Octave 5.1.0, main mesh (upper left), 1-face meshes (upper right), and 0-face meshes (bottom)

5.5 3d-orthotope meshed by orthotopes

In Listing 1, an `OrthMesh` object is built under Octave 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.



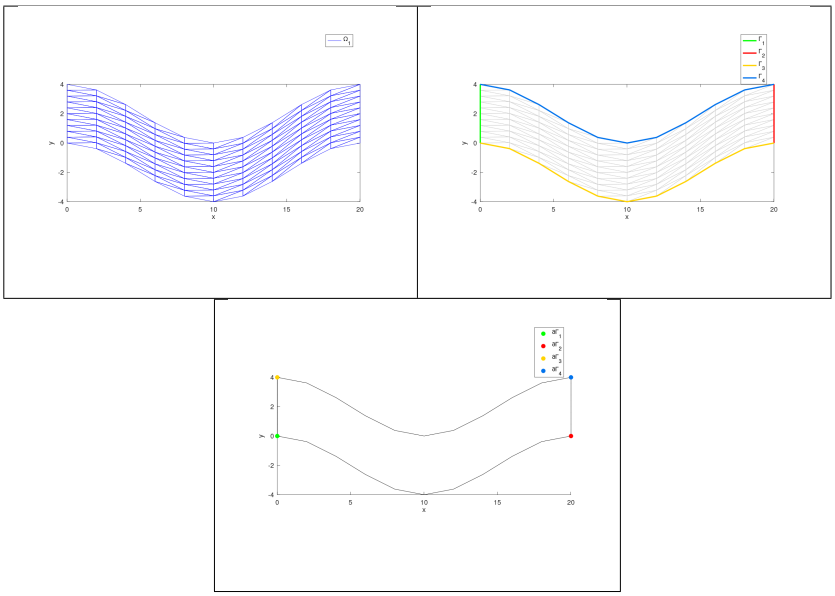
```
Oh=fc_hypermesh.OrthMesh(3,[10,5,10],'box',[-1,1;0,1;0,2], 'type','orthotope');
% plot the main mesh
figure(1)
Oh.plotmesh('legend',true)
axis equal;xlabel('x');ylabel('y');zlabel('z')
% plot the 2-face meshes
figure(2)
Oh.plotmesh('m',2,'legend',true)
axis equal;axis off
% plot the 1-face meshes
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 14: 3D orthotope `OrthMesh` object with Octave 5.1.0, main mesh (upper left), 2-face meshes (upper right), 1-face meshes (bottom left) and 0-face meshes (bottom right)

5.6 Mapping of a 2d-orthotope meshed by simplices

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

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



```
Oh=fc_hypermesh.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)
```

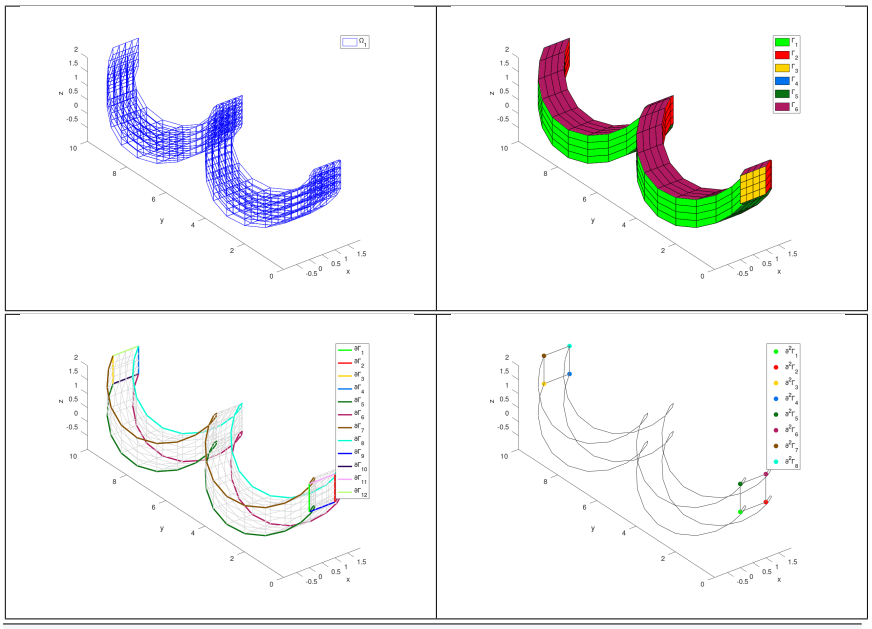
Listing 15: Mapping of a 2D simplicial `OrthMesh` object with Octave 5.1.0, main mesh (upper left), 1-face meshes (upper right), and 0-face meshes (bottom)

5.7 3d-orthotope meshed 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=fc_hypermesh.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 16: Mapping of a 3D orthotope `OrthMesh` object with Octave 5.1.0, main mesh (upper left), 2-face meshes (upper right), 1-face meshes (bottom left) and 0-face meshes (bottom right)

6 Benchmarking

6.1 `fc_hypermesh.bench` function

The `fc_hypermesh.bench` function can be used to obtain computationnal times of the `OrthMesh` constructor.

Syntaxe

```
fc_hypermesh.bench(d,LN)
fc_hypermesh.bench(d,LN, key,value, ...)
```

Description

`fc_bench.bench(d,LN)`

displays computationnal times of the `OrthMesh` constructor tessellated with simplices (default) as follows

```
ts=tic();Oh=fc_hypermesh.OrthMesh(d,LN(:,i);tcpu=toc(ts);
```

for each `i` in `1:size(LN,2)`.

- `d`, space dimension.
- `LN` is either a 1-by-n or a `d`-by-n array of positive integers.

`fc_bench.bench(d,LN, key,value, ...)`

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

- `'type'` : string `'simplex'` (default) or `'orthotope'` to specify kind of tessellation.
- `'order'` : positive integer (1 by default) to specify order of the mesh elements.
- `'box'` : `d`-by-2 array of double

6.2 Examples

Listing 17: : Computationnal times of `OrthMesh` constructor in dimension `d=3` where mesh elements are 1-order simplices

```
fc_hypermesh.bench(3,25:25:175,'box',[-1 1;-1 1;-1 1], 'type','simplex')
```

Output

```
# BENCH in dimension 3 with simplex mesh
#d: 3
#order: 1
#type: simplex
#box: [ -1 1; -1 1; -1 1]
#desc:     N        nq       nme  time(s)
          25     17576     93750   0.746
          50    132651    750000   0.922
          75    438976   2531250   1.747
         100   1030301   6000000   3.232
         125   2000376  11718750   5.668
         150   3442951  20250000   9.238
         175   5451776  32156250  14.913
```

Listing 18: : Computationnal times of `OrthMesh` constructor in dimension `d=3` where mesh elements are 3-order simplices

```
fc_hypermesh.bench(3,10:10:100,'box',[-1 1;-1 1;-1 1], ...
    'type','simplex','order',3)
```

Output

```
# BENCH in dimension 3 with simplex mesh
#d: 3
#order: 3
#type: simplex
#box: [ -1 1; -1 1; -1 1]
#desc:     N        nq       nme  time(s)
          10     29791      6000   0.700
          20    226981     48000   0.779
          30    753571    162000   0.937
          40   1771561    384000   1.200
          50   3442951    750000   1.732
          60   5929741   1296000   3.801
          70   9393931   2058000   6.122
          80  13997521   3072000   8.805
          90  19902511   4374000  12.334
         100  27270901   6000000  16.679
```

Listing 19: : Computationnal times of `OrthMesh` constructor in dimension `d=5` where mesh elements are 1-order orthotopes

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

Output

```
# BENCH in dimension 5 with orthotope mesh
#d: 5
#order: 1
#type: orthotope
#box: [ -1 1; -1 1; -1 1; -1 1; -1 1]
#desc:     N        nq       nme  time(s)
           5      7776      3125   4.474
          10    161051    100000   4.571
          15   1048576    759375   5.510
          20   4084101   3200000   7.703
          25  11881376   9765625  13.513
          27  17210368  14348907  17.413
```

Listing 20: : Computationnal times of `OrthMesh` constructor in dimension `d=5` where mesh elements are 3-order orthotopes

```
fc_hypermesh.bench(5,2:2:12,'type','orthotope','order',3, 'box',[-1 1;-1 1;-1 ...
    1;-1 1;-1 1])
```

Output

```
# BENCH in dimension 5 with orthotope mesh
#d: 5
#order: 3
#type: orthotope
#box: [ -1 1; -1 1; -1 1; -1 1; -1 1]
#desc:     N        nq       nme  time(s)
           2     16807        32   4.752
           4    371293      1024   5.017
           6   2476099      7776   5.899
           8   9765625     32768   8.809
          10  28629151    100000  15.506
          12  69343957    248832  29.597
```

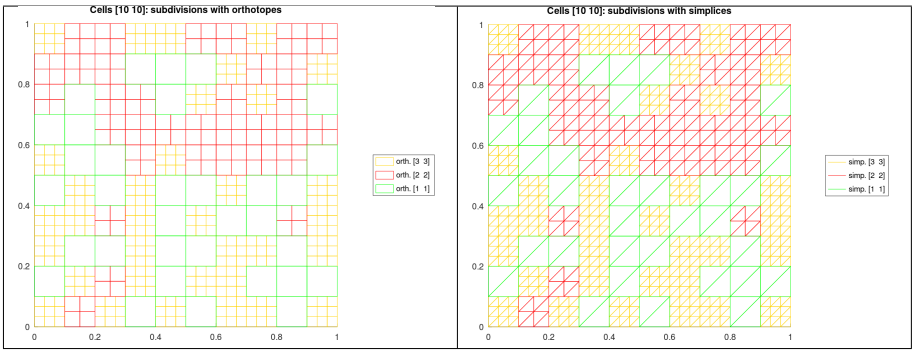


Figure 9: regular refinement of a 2D-grid with orthotopes (left) and simplices (right)

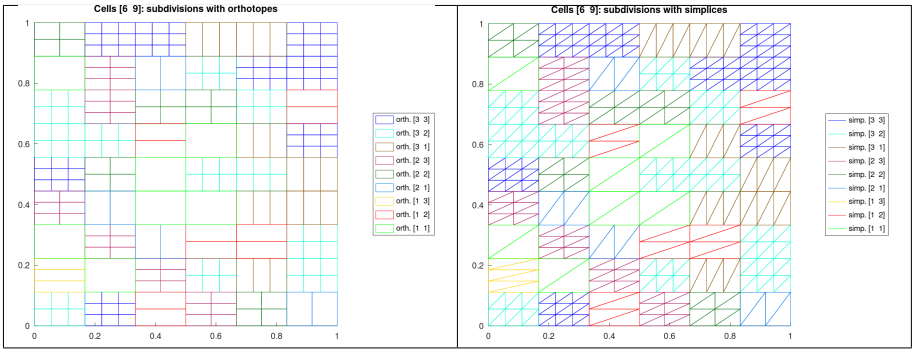


Figure 10: no regular refinement of a 2D-grid with orthotopes (left) and simplices (right)

7 Mesh refinement

7.1 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.

In Figure 9, 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 10, 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 11, 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.

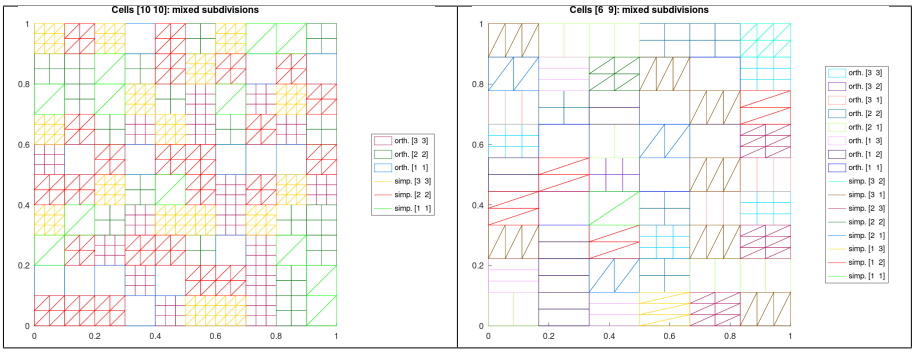


Figure 11: Regular (left) and not regular (right) refinement of a 2D-grid with orthotopes and simplices

7.1.1 Refinement function

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

```
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 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 ($p = \texttt{d} + 1$ for simplices and $p = 2^{\texttt{d}}$ 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.

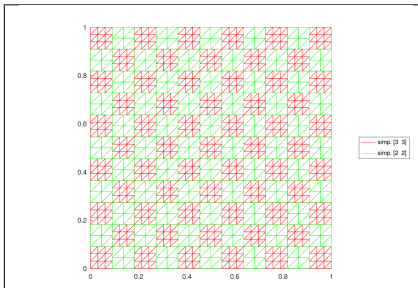
`ndiscells` is an 1-by-`Oh.Mesh.nme` array (for regular refinement) or a `d`-by-`Oh.Mesh.nme` array. `ndiscells(:,k)` are the numbers of discretisation in each axis direction. The output is a cells array of *subcells structure* where a all subcells `type` are `'orthotope'`

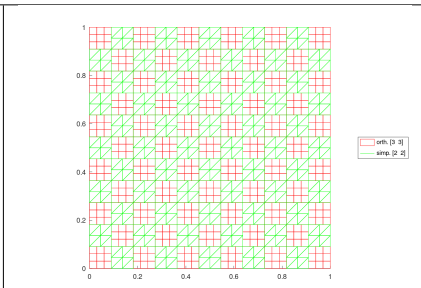
```
scs = fc_hypermesh.refinement.refine(Oh,ndiscells,type)
```

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=fc_hypermesh.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,'simplex');
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 21: 2D orthotope `OrthMesh` refinement with Octave 5.1.0, figure 1 (left) and figure 2 (right)

More examples are provided by scripts:

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

Appendices

A Memory consuming

Beware when using theses codes of memory consuming : the number of points nq and the number of elements quickly increase according to the space dimension d. When meshing a d-orthotope with p-order mesh elements by taking N intervals in each space direction, we have a number of nodes given by

$$\texttt{nq} = (\texttt{p} * N + 1)^{\texttt{d}}, \quad \text{for both tessellation and triangulation}$$

and the number of mesh elements is

$$\begin{array}{rcll} \texttt{nme} & = & N^d, & \text{for tessellation by orthotopes} \\ \texttt{nme} & = & d!N^d, & \text{for tessellation by simplices.} \end{array} .$$

If the array q is stored as *double* (8 bytes) then

$$\text{mem. size of } \texttt{q} = \texttt{d} \times \texttt{nq} \times 8 \text{ bytes}$$

and if the array me as *int* (4 bytes) then

$$\text{mem. size of } \texttt{me} = \begin{cases} (\texttt{p}+1)^{\texttt{d}} \times \texttt{nme} \times 4 \text{ bytes} & \text{(tessellation by orthotopes)} \\ \frac{(\texttt{d}+\texttt{p})!}{\texttt{d}!\texttt{p}!} \times \texttt{nme} \times 4 \text{ bytes} & \text{(tessellation by simplices)} \end{cases}$$

With $N = 10$, $\texttt{d} \in [\![1, 8]\!]$ and order $\texttt{p} = 1$, the values of nq and nme are given in Table 5. The memory usage for the corresponding array q and array me is available in Table 6. In Table **??**, the memory usage for the array q and array me for $N = 10$, $\texttt{d} = 3$ and order $\texttt{p} \in [\![1, 8]\!]$.

d	nq $= (N+1)^{\texttt{d}}$	nme $= N^{\texttt{d}}$ (orthotopes)	nme $= \texttt{d}!N^{\texttt{d}}$ (simplices)
1	??	??	??
2	??	??	??
3	??	??	??
4	??	??	??
5	??	??	??
6	??	??	??
7	??	??	??
8	??	??	??

Table 5: Number of vertices nq and number of elements nme for the tessellation of an orthotope by orthotopes and by simplices according to the space dimension d and with $N = 10$.

d	q	me (orthotopes)	me (simplices)
1	??	??	??
2	??	??	??
3	??	??	??
4	??	??	??
5	??	??	??
6	??	??	??
7	??	??	??
8	??	??	??

Table 6: Memory usage of the array q and the array me for the tessellation of an orthotope by 1-order orthotopes and by 1-order simplices according to the space dimension d and with $N = 10$.

d	q	me (orthotopes)	me (simplices)
1	??	??	??
2	??	??	??
3	??	??	??
4	??	??	??
5	??	??	??
6	??	??	??
7	??	??	??
8	??	??	??

Table 7: Memory usage of the array q and the array me for the tessellation of an orthotope by 3-order orthotopes and by 3-order simplices according to the space dimension d and with $N = 10$.

B Low level codes

B.1 Class object EltMesh

The elementary mesh class object called 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 (m = d for the main mesh), tessellated with m-simplices or m-orthotopes.
- order, order p of the mesh elements
- type, 0 for simplicial mesh or 1 for orthotope mesh
- nq, number of nodes
- q, nodes array of dimension d-by-nq
- nme, number of mesh elements

- me, connectivity array of dimension $\frac{(d+p)!}{d!p!}$-by-nme for p-order simplices elements or $(p+1)^d$-by-nme for p-order 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)

B.1.1 Constructor

```
Eh = fc_hypermesh.EltMesh(m,q,me)
Eh = fc_hypermesh.EltMesh(m,q,me, key,value,...)
```

Description

```
Eh = fc_hypermesh.EltMesh(m,q,me)
```

Genrerates the EltMesh object Eh which contains a 1-order m-simplicial mesh by default. The space dimension d is size(q,1).

- m, kind of mesh: tessellated with m-simplices.
- q is the nodes array of dimension d-by-nq.
- me is the connectivity array of dimension (d+1)-by-nme corresponding to 1-order simplices elements.

Listing 22: : EltMesh constructor in dimension d=2 (simplicial mesh)

```
[q,me]=fc_hypermesh.CG.TessSim(10^3*ones(1,2));
Eh=fc_hypermesh.EltMesh(2,q,me)
```

Output

```
Eh =

EltMesh object
   type : simplex
  order : 1
      d : 2
      m : 2
      q : (2,1002001)
     me : (3,2000000)
```

```
Eh = fc_hypermesh.EltMesh(m,q,me, key,value, ...)
```

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

- 'type' : used to select the kind of elements used. The default value is 'simplex' and otherwise 'orthotope' can be used.
- 'order' : gives the order of the mesh elements (default is 1).

Listing 23: : `EltMesh` constructor in dimension `d=3` (orthotope mesh)

```
[q,me]=fc_hypermesh.CG.TessHyp(3:5,2);
Eh=fc_hypermesh.EltMesh(3,q,me,'type','orthotope','order',2)
```

Output

```
Eh =

EltMesh object
   type : orthotope
  order : 2
      d : 3
      m : 3
      q : (3,693)
     me : (27,60)
```

B.1.2 `plotmesh` method

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

Syntaxe

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

Description

`obj.plotmesh()`

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

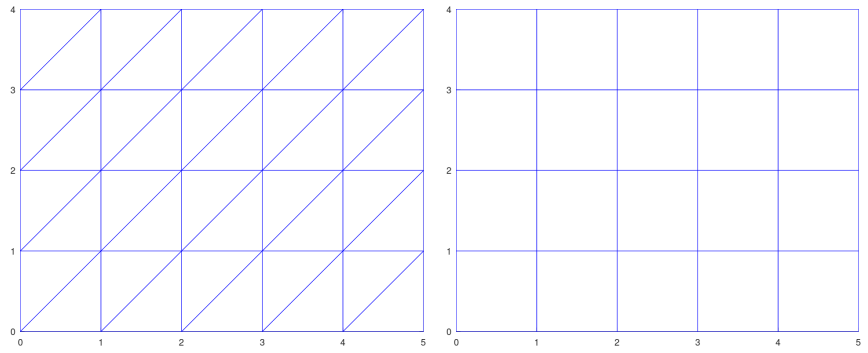
Some optional `key`/`value` pairs arguments are available with `key`:

- `'color'` : use to specify the color of the mesh. Otherwise `obj.color` is used.

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;



```
[q,me]=fc_hypermesh.CG.TessSim([5,4]);
Eh1=fc_hypermesh.EltMesh(2,q,me);
figure(1)
Eh1.plotmesh()
axis image
[q,me]=fc_hypermesh.CG.TessHyp([5,4],2);
Eh2=fc_hypermesh.EltMesh(2,q,me,'order',2,'type','orthotope');
figure(2)
Eh2.plotmesh()
axis image
```

Listing 24: `plotmesh` method of a 2D `EltMesh` objects , `figure(1)` (left) and `figure(2)` (right)

B.1.3 `plotnodes` method

The `plotnodes()` member function can be used to represent nodes of the mesh given by an `EltMesh` object if the space dimension `d` is less than or equal to 3.

Syntaxe

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

Description

`obj.plotnodes()`

Uses `scatter` function (1D and 2D) and `scatter3` function (3D) to represent nodes of the mesh as points. Vertices of the mesh elements are also nodes and they are distinguishable from others nodes.

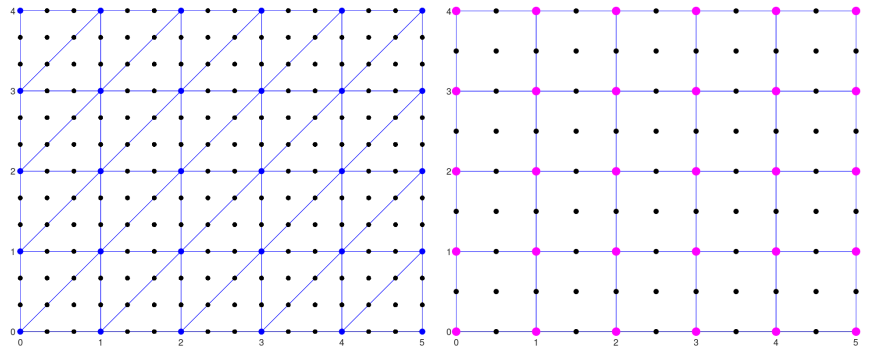
`obj.plotnodes(key, value, ...)`

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

- `'vcolor'` : use to specify the point color for the mesh vertices. Default is `obj.color` .
- `'vsize'` : use to specify the point size for the mesh vertices. Default is `40` .

- `'ncolor'` : use to specify the color of the nodes (not vertices) of the mesh elements. Default is `'k'` (ie. black).
- `'nsize'` : use to specify the size of the nodes (not vertices) of the mesh elements. Default is `30` .

Other `key/value` pairs arguments can be used: they are those of the `scatter` and `scatter3` function used.



```
[q,me]=fc_hypermesh.CG.TessSim([5,4],3);
Eh1=fc_hypermesh.EltMesh(2,q,me,'order',3);
figure(1)
Eh1.plotmesh()
hold on
Eh1.plotnodes()
axis image
[q,me]=fc_hypermesh.CG.TessHyp([5,4],2);
Eh2=fc_hypermesh.EltMesh(2,q,me,'order',2,'type','orthotope');
figure(2)
Eh2.plotmesh()
hold on
Eh2.plotnodes('vcolor','m','vsize',80)
axis image
```

Listing 25: `plotnodes` method of a 2D `EltMesh` objects , `figure(1)` (left) and `figure(2)` (right)

B.1.4 `plotnodesNumber` method

The `plotnodesNumber()` member function can be used to display node numbers of the mesh given by an `EltMesh` object if the space dimension `d` is less than or equal to 3.

Syntaxe

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

Description

`obj.plotnodes()`

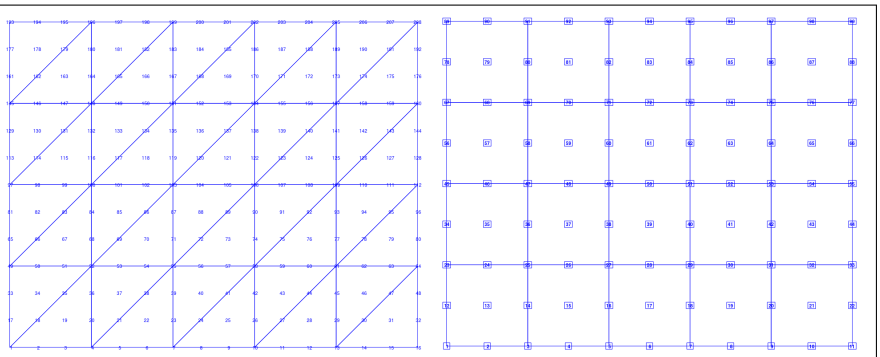
Uses `fc_hypermesh.plotnodesNumber` function to represent node numbers.

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

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

- 'BackgroundColor' : use to specify text background color. Default is 'w' (i.e. white).
- 'margin' : use to specify the margin. Default is 1 .
- 'HorizontalAlignment' : Default is 'center' .
- 'VerticalAlignment' : Default is 'middle' .
- 'clipping' : Default is 'on' .

Other key/value pairs arguments can be used: they are those of the text function used.



```
[q,me]=fc_hypermesh.CG.TessSim([5,4],3);
Eh1=fc_hypermesh.EltMesh(2,q,me,'order',3);
figure(1)
Eh1.plotmesh()
hold on
Eh1.plotnodesNumber()
axis image
axis off
[q,me]=fc_hypermesh.CG.TessHyp([5,4],2);
Eh2=fc_hypermesh.EltMesh(2,q,me,'order',2,'type','orthotope');
figure(2)
Eh2.plotmesh()
hold on
Eh2.plotnodesNumber('color','b','EdgeColor','b','FontWeight','bold')
axis image
axis off
```

Listing 26: plotnodesNumber method of a 2D EltMesh objects , figure(1) (left) and figure(2) (right)

B.1.5 ploteltsNumber method

The ploteltsNumber() member function can be used to display element numbers of the mesh given by an EltMesh object if the space dimension d is less than or equal to 3.

Syntaxe

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

Description

`obj.ploteltsNumber()`

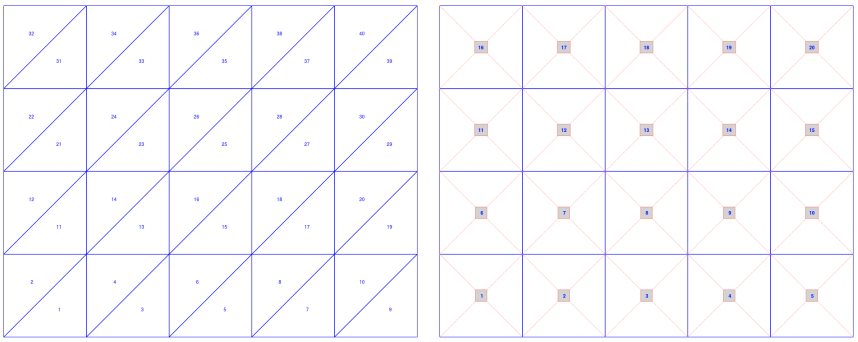
Uses `fc_hypermesh.ploteElementsNumber` function to represent node numbers.

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

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

- `'Color'` : use to specify text color. Default is `'r'` (i.e. red).
- `'BackgroundColor'` : use to specify text background color. Default is `'w'` (i.e. white).
- `'margin'` : use to specify the margin. Default is `1` .
- `'HorizontalAlignment'` : Default is `'center'` .
- `'VerticalAlignment'` : Default is `'middle'` .
- `'clipping'` : Default is `'on'` .
- `'vLineColor'` : Draw lines beetween vertices and barycenter of the mesh elements. Default is `''` (no lines).
- `'vLineStyle'` : Select lines type. Default is `':'` (dotted lines).
- `'vLineWidth'` : Set lines witdh. Default is `0.5` .

Other `key`/`value` pairs arguments can be used: they are those of the `text` function used.



```
[q,me]=fc_hypermesh.CG.TessSim([5,4],3);
Eh1=fc_hypermesh.EltMesh(2,q,me,'order',3);
figure(1)
Eh1.plotmesh()
hold on
Eh1.ploteltsNumber()
axis image
axis off
[q,me]=fc_hypermesh.CG.TessHyp([5,4],2);
Eh2=fc_hypermesh.EltMesh(2,q,me,'order',2,'type','orthotope');
figure(2)
Eh2.plotmesh()
hold on
Eh2.ploteltsNumber('color','b','EdgeColor','salmon','FontWeight','bold', ...
    'vLineColor','salmon','BackgroundColor','LightGrey' )
axis image
axis off
```

Listing 27: `ploteltsNumber` method of a 2D `EltMesh` objects , `figure(1)` (left) and `figure(2)` (right)

H References

[1] F. Cuvelier. Vectorized algorithms for regular and conforming tessellations of d-orthotopes and their faces with high-order orthotopes or simplicial elements. working paper or preprint, March 2019.