fc-hypermesh Octave package, User’s Guide * version 1.0.2
Francois Cuvelier

To cite this version:
Francois Cuvelier. fc-hypermesh Octave package, User’s Guide * version 1.0.2. 2019. hal-02425536

HAL Id: hal-02425536
https://sorbonne-paris-nord.hal.science/hal-02425536
Preprint submitted on 30 Dec 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
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 `fc_hypermesh` package uses Octave objects and is provided with meshes visualisation tools for dimension less than or equal to 3.
Introduction

The fc_hypermesh package contains a simple class object OrthMesh which permits, in any dimension \( d \geq 1 \), to obtain conforming mesh of a \( d \)-orthotope tessellated with \( p \)-order simplices or \( p \)-order orthotopes. Corresponding \( m \)-faces, \( 0 \leq m < d \) of the mesh are also provided. The number of \( m \)-faces of a \( d \)-orthotope is

\[
E_m^d \overset{\text{def}}{=} 2^{d-m} \left( \begin{array}{c} d \\ m \end{array} \right) \quad \text{where} \quad \left( \begin{array}{c} d \\ m \end{array} \right) = \frac{d!}{m!(d-m)!}
\]

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

<table>
<thead>
<tr>
<th>d</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[diagram]</td>
<td>[diagram]</td>
<td>[diagram]</td>
<td>[diagram]</td>
</tr>
<tr>
<td>2</td>
<td>[diagram]</td>
<td>[diagram]</td>
<td>[diagram]</td>
<td>[diagram]</td>
</tr>
<tr>
<td>3</td>
<td>[diagram]</td>
<td>[diagram]</td>
<td>[diagram]</td>
<td>[diagram]</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>d</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[diagram]</td>
<td>[diagram]</td>
<td>[diagram]</td>
<td>[diagram]</td>
</tr>
<tr>
<td>2</td>
<td>[diagram]</td>
<td>[diagram]</td>
<td>[diagram]</td>
<td>[diagram]</td>
</tr>
<tr>
<td>3</td>
<td>[diagram]</td>
<td>[diagram]</td>
<td>[diagram]</td>
<td>[diagram]</td>
</tr>
</tbody>
</table>

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 these meshes, all the associated 2-faces meshes are represented in Figure 2 and Figure 4.
Figure 1: Tessellation 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 tessellation samples of the Figure 1.
Figure 3: Tessellation 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 tessellation 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: Tessellation 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 tessellation 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: Tessellation 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 tessellation 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 $d \leq 3$.

This package was tested on various OS with Octave releases:
<table>
<thead>
<tr>
<th>Operating system</th>
<th>4.2.0</th>
<th>4.2.1</th>
<th>4.2.2</th>
<th>4.4.0</th>
<th>4.4.1</th>
<th>5.1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS 7.7.1908</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Debian 9.11</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Fedora 29</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>OpenSUSE Leap 15.0</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Ubuntu 18.04.3 LTS</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MacOS High Sierra 10.13.6</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MacOS Mojave 10.14.4</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Windows 10 (1909)</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

It is not compatible with Octave releases prior to 4.2.0.

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

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

### 3 All-in-one installation

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

```plaintext
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:

```plaintext
# mkdir -p ~/Octave/packages
# cd ~/Octave/packages
# wget http://www.math.univ-paris13.fr/~cuvelier/software/codes/Octave/fc-hypermesh/1.0.2/ofc_hypermesh_install.m

Then in a Octave terminal run the following commands

```plaintext
>> 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-1.0.2/
cfgure_loc.m ...
3- Using packages :
  -> fc-tools : 0.0.29
  -> fc-bench : 0.1.1
*** Using instructions
To use the `<fc-hypermesh>` package:
  addpath('/home/Octave/packages/fc-hypermesh-full/fc_hypermesh-1.0.2')
  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 `/home/ Octave/packages/fc-hypermesh-full` and, for each Octave
session, one has to set the package by:

```octave
>> addpath('/home/ Octave/packages/fc-hypermesh-full/ofc-hypermesh-1.0.2')
>> fc_hypermesh.init()
```

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

4 Installation via pkg command

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

```bash
wget http://www.math.univ-paris13.fr/~cuvelier/software/codes/Octave/fc-tools/0.0.29/fc-tools-0.0.29.tar.gz
wget http://www.math.univ-paris13.fr/~cuvelier/software/codes/Octave/fc-bench/0.1.1/fc-bench-0.1.1.tar.gz
wget http://www.math.univ-paris13.fr/~cuvelier/software/codes/Octave/fc-hypermesh/1.0.2/fc-hypermesh-1.0.2.tar.gz
```

- Under Octave :

```octave
>> pkg install fc-tools-0.0.29.tar.gz
>> pkg install fc-bench-0.1.1.tar.gz
>> pkg install fc-hypermesh-1.0.2.tar.gz
```

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

```octave
pkg install http://www.math.univ-paris13.fr/~cuvelier/software/codes/Octave/fc-tools/0.0.29/fc-tools-0.0.29.tar.gz
pkg install http://www.math.univ-paris13.fr/~cuvelier/software/codes/Octave/fc-bench/0.1.1/fc-bench-0.1.1.tar.gz
pkg install http://www.math.univ-paris13.fr/~cuvelier/software/codes/Octave/fc-hypermesh/1.0.2/fc-hypermesh-1.0.2.tar.gz
```

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

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

- To try the package, one can launch a demo:
For uninstalling the package, just do in an Octave session:

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

## 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 6-dimensional orthotope with 1-order simplices by taking $N = 10$ intervals in each space direction, gives an `OrthMesh` object using 20 GB in memory. With 3-order simplices, the `OrthMesh` object use 241 GB in memory!

The memory usage for a $d$-dimensional `OrthMesh` object by taking $N = 10$ intervals in each space direction is given in Table 3 for 1-order elements and in Table 4 for 3-order elements. One can refer to Appendix A for more details.

### 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 $N = 10$.

<table>
<thead>
<tr>
<th>$d$</th>
<th>OrthMesh (orthotopes)</th>
<th>OrthMesh (simplices)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8 B</td>
<td>80 B</td>
</tr>
<tr>
<td>2</td>
<td>688 B</td>
<td>2 KB</td>
</tr>
<tr>
<td>3</td>
<td>23 KB</td>
<td>96 KB</td>
</tr>
<tr>
<td>4</td>
<td>601 KB</td>
<td>4 MB</td>
</tr>
<tr>
<td>5</td>
<td>13 MB</td>
<td>288 MB</td>
</tr>
<tr>
<td>6</td>
<td>308 MB</td>
<td>20 GB</td>
</tr>
<tr>
<td>7</td>
<td>6 GB</td>
<td>1 TB</td>
</tr>
<tr>
<td>8</td>
<td>147 GB</td>
<td>145 TB</td>
</tr>
</tbody>
</table>

### Table 4: Memory usage of `OrthMesh` object for the tessellation of an orthotope by 3-order orthotopes and by 3-order simplices according to the space dimension $d$ and with $N = 10$.

<table>
<thead>
<tr>
<th>$d$</th>
<th>OrthMesh (orthotopes)</th>
<th>OrthMesh (simplices)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8 B</td>
<td>160 B</td>
</tr>
<tr>
<td>2</td>
<td>1 KB</td>
<td>8 KB</td>
</tr>
<tr>
<td>3</td>
<td>135 KB</td>
<td>480 KB</td>
</tr>
<tr>
<td>4</td>
<td>8 MB</td>
<td>33 MB</td>
</tr>
<tr>
<td>5</td>
<td>438 MB</td>
<td>2 GB</td>
</tr>
<tr>
<td>6</td>
<td>21 GB</td>
<td>241 GB</td>
</tr>
<tr>
<td>7</td>
<td>968 GB</td>
<td>24 TB</td>
</tr>
<tr>
<td>8</td>
<td>42 TB</td>
<td>2 661 TB</td>
</tr>
</tbody>
</table>
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 $[a_1, b_1] \times \cdots \times [a_d, b_d]$. The class object OrthMesh corresponding to this $d$-orthotope contains the main mesh and all its $m$-face meshes, $0 \leq m < d$. 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 $(d-1)$-face meshes stored as EltMesh objects, Faces{2} is an array of all the $(d-2)$-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

```matlab
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)**

Generates 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)**

Generates 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.

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

```python
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.

```python
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).

```python
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 $[a_1, b_1] \times \ldots \times [a_d, b_d]$ by setting `value` as an $d$-by-2 array such that $a_i = \text{value}(i,1)$ and $b_i = \text{value}(i,2)$.
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
```

- `$m_{\text{min}}$`: used to only build the $m$-Faces for $m$ in $[m_{\text{min}}, d]$. Default value is 0.

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

```
Oh = fc_hypermesh.OrthMesh(3,10, 'm_{\text{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
It's a 3 dimensional mesh of the unit cube tessellated 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

\[
\begin{aligned}
\text{Th} &= \text{Oh.}\text{Mesh} \\
\end{aligned}
\]

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

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

\[
\text{Th.q}(:,\text{Fh.toGlobal})==\text{Fh.q}
\]

or more generally, for all \(m \in [0, Oh.d]\) and for all \(k \in [1, E^d_m]\)

\[
Oh.\text{Mesh.q}(:,\text{Oh.}\text{Faces(Oh.d-m)}(k).\text{toGlobal})==\text{Oh.}\text{Faces(Oh.d-m)}(k).q
\]

where \(E^d_m\) 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

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

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)
**Description**

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.

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. ($0 \leq m \leq d$)
- `'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.
5.1.5 plotnodesNumber method

The plotnodesNumber() member function can be used to display nodes index/number of the mesh given by an OrthMesh object if the space dimension $d$ is less than 19.
or equal to 3.

Syntaxe

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

Description

```python
obj.plotnodesNumber()
```
Uses `fc_hypermesh.plotnodesNumber` function to represent node numbers.

```python
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 ≤ m ≤ d)
- `'labels'`: plots 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.
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**

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

**Description**

- `obj.ploteltsNumber()`
  
  Uses `fc_hypermesh.plotElementsNumber` 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. ($0 \leq m \leq d$)
  - `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 between vertices and barycenter of the mesh elements. Default is '' (no lines).
• 'vLineStyle': Select lines type. Default is ':' (dotted lines).
• 'vLineWidth': Set lines width. 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 \( \mathcal{O}_n \) 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.

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 \( \mathbf{N}=[10,5,10] \). The main mesh and all the \( m \)-face meshes of the resulting object are plotted.

### 5.3 3d-orthotope meshed by simplices

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

Listing 1: 2D simplicial `OrthMesh` object with Octave 5.1.0, main mesh (upper left), 1-face meshes (upper right), and 0-face meshes (bottom)
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.
In Listing 13, an `OrthMesh` object is built under Octave for the orthotope $[-1, 1] \times [0, 1] \times [0, 2]$ with orthotope elements and $\mathbf{N} = [10, 5, 10]$. The main mesh and all the $\mathbf{m}$-face meshes of the resulting object are plotted.
5.6 Mapping of a 2d-orthotope meshed by simplices

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

\[
[0, 1] \times [0, 1] \longrightarrow \mathbb{R}^2
\]

\[
\begin{pmatrix} x \\ y \end{pmatrix} \longrightarrow F(x, y) = \begin{pmatrix} 20x \\ 2(2y - 1 + \cos(2\pi x)) \end{pmatrix}
\]
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.

\[ [0, 1] \times [0, 1] \times [0, 1] \rightarrow \mathbb{R}^2 \]

\[
\begin{pmatrix}
  x \\
  y \\
  z
\end{pmatrix}
\rightarrow
F(x, y, z) =
\begin{pmatrix}
  x + \sin(4\pi y) \\
  10y \\
  z + \cos(4\pi y)
\end{pmatrix}
\]
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 computational times of the OrthMesh constructor.

Syntaxe

\[
\begin{align*}
&\text{fc_hypermesh.bench}(d, LN) \\
&\text{fc_hypermesh.bench}(d, LN, \text{key}, \text{value}, \ldots)
\end{align*}
\]
Description

```plaintext
fc_bench.bench(d,LN)
```

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

```plaintext
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.

```plaintext
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: Computational times of OrthMesh constructor in dimension d=3 where mesh elements are 1-order simplices

```plaintext
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  0.391
50  132651  750000  0.432
75  438976  2531250  0.662
100 1030301  6000000  0.977
125 2000376 11718750 1.512
150 3442951 20250000 2.426
175 5451776 32156250 4.022
```
Listing 18: Computational times of `OrthMesh` constructor in dimension d=3 where mesh elements are 3-order simplices

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

Output

<table>
<thead>
<tr>
<th>N</th>
<th>nq</th>
<th>nme</th>
<th>time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>29791</td>
<td>6000</td>
<td>0.351</td>
</tr>
<tr>
<td>20</td>
<td>226981</td>
<td>48000</td>
<td>0.470</td>
</tr>
<tr>
<td>30</td>
<td>75371</td>
<td>16200</td>
<td>0.558</td>
</tr>
<tr>
<td>40</td>
<td>171561</td>
<td>38400</td>
<td>0.719</td>
</tr>
<tr>
<td>50</td>
<td>342561</td>
<td>75000</td>
<td>1.001</td>
</tr>
<tr>
<td>60</td>
<td>592741</td>
<td>129600</td>
<td>1.465</td>
</tr>
<tr>
<td>70</td>
<td>939391</td>
<td>205800</td>
<td>2.094</td>
</tr>
<tr>
<td>80</td>
<td>1399751</td>
<td>307200</td>
<td>2.928</td>
</tr>
<tr>
<td>90</td>
<td>1990251</td>
<td>437400</td>
<td>4.053</td>
</tr>
<tr>
<td>100</td>
<td>2727091</td>
<td>600000</td>
<td>5.399</td>
</tr>
</tbody>
</table>

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

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

Output

<table>
<thead>
<tr>
<th>N</th>
<th>nq</th>
<th>nme</th>
<th>time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>7776</td>
<td>3125</td>
<td>2.257</td>
</tr>
<tr>
<td>10</td>
<td>161051</td>
<td>100000</td>
<td>2.397</td>
</tr>
<tr>
<td>15</td>
<td>1048576</td>
<td>75000</td>
<td>3.133</td>
</tr>
<tr>
<td>20</td>
<td>4084101</td>
<td>3200000</td>
<td>4.565</td>
</tr>
<tr>
<td>25</td>
<td>11881376</td>
<td>9765625</td>
<td>8.668</td>
</tr>
<tr>
<td>27</td>
<td>17210368</td>
<td>14348907</td>
<td>11.299</td>
</tr>
</tbody>
</table>

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

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

Output

<table>
<thead>
<tr>
<th>N</th>
<th>nq</th>
<th>nme</th>
<th>time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>16807</td>
<td>32</td>
<td>2.297</td>
</tr>
<tr>
<td>4</td>
<td>371293</td>
<td>1024</td>
<td>2.600</td>
</tr>
<tr>
<td>6</td>
<td>2476099</td>
<td>7776</td>
<td>3.164</td>
</tr>
<tr>
<td>8</td>
<td>9766265</td>
<td>32768</td>
<td>4.992</td>
</tr>
<tr>
<td>10</td>
<td>28829151</td>
<td>100000</td>
<td>9.158</td>
</tr>
<tr>
<td>12</td>
<td>69343957</td>
<td>248832</td>
<td>18.138</td>
</tr>
</tbody>
</table>
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 $\text{orth}[3 \times 3]$ and this same cell subdivided in 3-by-3 simplices (in fact $3 \times 3 \times 2$ simplices) is denoted by $\text{simp}[3 \times 3 \times 2]$. 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.
7.1.1 Refinement function

The refinement of an OrthMesh object whose elements are orthotopes is obtained by using the \texttt{fc_hypermesh.refinement.refine} Octave function.

\begin{verbatim}
scs = refine(Oh,ndiscells)
scs = refine(Oh,ndiscells,type)
scs = refine(Oh,ndiscells,types)
\end{verbatim}

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 \textit{subcells structure}) has the fields

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

\section*{Description}

\begin{verbatim}
scs = fc_hypermesh.refinement.refine(Oh,ndiscells)
\end{verbatim}

The first input \textit{Oh} is an \texttt{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'

```matlab
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'

```matlab
[scesimp,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.

```matlab
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;
sc = fc_hypermesh.refinement.refine(Oh.ndiscells,'simplex');
figure(1)
fc_hypermesh.refinement.plotsubcells(sc)
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 $n_q$ 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

$$n_q = (p \times N + 1)^d,$$

for both tessellation and triangulation

and the number of mesh elements is

$$n_{me} = \begin{cases} N^d, & \text{for tessellation by orthotopes} \\ d!N^d, & \text{for tessellation by simplices}. \end{cases}$$

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

mem. size of $q = d \times n_q \times 8$ bytes

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

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

With $N = 10$, $d \in [1, 8]$ and order $p = 1$, the values of $n_q$ and $n_{me}$ 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 $N = 10$, $d = 3$ and order $p \in [1, 8]$.

<table>
<thead>
<tr>
<th>$d$</th>
<th>$n_q = (N + 1)^d$</th>
<th>$n_{me} = N^d$ (orthotopes)</th>
<th>$n_{me} = d!N^d$ (simplices)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>121</td>
<td>100</td>
<td>200</td>
</tr>
<tr>
<td>3</td>
<td>1 331</td>
<td>1000</td>
<td>6 000</td>
</tr>
<tr>
<td>4</td>
<td>14 641</td>
<td>10 000</td>
<td>240 000</td>
</tr>
<tr>
<td>5</td>
<td>161 051</td>
<td>100 000</td>
<td>12 000 000</td>
</tr>
<tr>
<td>6</td>
<td>1 771 561</td>
<td>1 000 000</td>
<td>720 000 000</td>
</tr>
<tr>
<td>7</td>
<td>19 487 171</td>
<td>10 000 000</td>
<td>50 400 000 000</td>
</tr>
<tr>
<td>8</td>
<td>214 358 881</td>
<td>100 000 000</td>
<td>4 032 000 000 000</td>
</tr>
</tbody>
</table>

Table 5: Number of vertices $n_q$ and number of elements $n_{me}$ for the tessellation of an orthotope by orthotopes and by simplices according to the space dimension $d$ and with $N = 10$. 
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 \( N = 10 \).

<table>
<thead>
<tr>
<th>( d )</th>
<th>( q ) (orthotopes)</th>
<th>( me ) (simplices)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>88 B</td>
<td>80 B</td>
</tr>
<tr>
<td>2</td>
<td>1 KB</td>
<td>1 KB</td>
</tr>
<tr>
<td>3</td>
<td>31 KB</td>
<td>32 KB</td>
</tr>
<tr>
<td>4</td>
<td>468 KB</td>
<td>640 KB</td>
</tr>
<tr>
<td>5</td>
<td>6 MB</td>
<td>12 MB</td>
</tr>
<tr>
<td>6</td>
<td>85 MB</td>
<td>256 MB</td>
</tr>
<tr>
<td>7</td>
<td>1 GB</td>
<td>5 GB</td>
</tr>
<tr>
<td>8</td>
<td>13 GB</td>
<td>102 GB</td>
</tr>
</tbody>
</table>

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 \( N = 10 \).

<table>
<thead>
<tr>
<th>( d )</th>
<th>( q ) (orthotopes)</th>
<th>( me ) (simplices)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>248 B</td>
<td>160 B</td>
</tr>
<tr>
<td>2</td>
<td>15 KB</td>
<td>6 KB</td>
</tr>
<tr>
<td>3</td>
<td>714 KB</td>
<td>256 KB</td>
</tr>
<tr>
<td>4</td>
<td>29 MB</td>
<td>10 MB</td>
</tr>
<tr>
<td>5</td>
<td>1 GB</td>
<td>409 MB</td>
</tr>
<tr>
<td>6</td>
<td>42 GB</td>
<td>16 GB</td>
</tr>
<tr>
<td>7</td>
<td>1 TB</td>
<td>655 GB</td>
</tr>
<tr>
<td>8</td>
<td>54 TB</td>
<td>26 TB</td>
</tr>
</tbody>
</table>

B Low level codes

B.1 Class object \texttt{EltMesh}

The elementary mesh class object called \texttt{EltMesh} is used to store only one mesh, the main mesh as well as any of the \( m \)-face meshes. This class \texttt{EltMesh} also simplifies (for me) the codes writing and its fields are the following:

- \( d \), space dimension
- \( m \), kind of mesh \((m = d \text{ for the main mesh})\), tessellated with \( m \)-simplices or \( m \)-orthotopes.
- \texttt{order}, order \( p \) of the mesh elements
- \texttt{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 \(\frac{(d+p)\times nme}{d+p}\) for \(p\)-order simplices elements or \((p+1)^{d}\)-by-\(nme\) for \(p\)-order orthotopes elements

• \texttt{toGlobal}, index array linking local array \(q\) to the one of the main mesh

• \texttt{label}, name/number of this elementary mesh

• \texttt{color}, color of this elementary mesh (for plotting purpose)

B.1.1 Constructor

\[
\begin{align*}
\texttt{Eh} &= \texttt{fc\_hypermesh.EltMesh}(m,q,me) \\
\texttt{Eh} &= \texttt{fc\_hypermesh.EltMesh}(m,q,me, key,value,\ldots)
\end{align*}
\]

Description

\[
\begin{align*}
\texttt{Eh} &= \texttt{fc\_hypermesh.EltMesh}(m,q,me)
\end{align*}
\]

Generates the \texttt{EltMesh} object \(\texttt{Eh}\) which contains a 1-order \(m\)-simplicial mesh by default. The space dimension \(d\) is \texttt{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 \((d+1)\times nme\) corresponding to 1-order simplices elements.

\[
\begin{align*}
\texttt{[q,me]} &= \texttt{fc\_hypermesh.CG\_TessSim}(10^3 \times \texttt{ones}(1,2)); \\
\texttt{Eh} &= \texttt{fc\_hypermesh.EltMesh}(2,q,me)
\end{align*}
\]

Output

\[
\begin{align*}
\texttt{Eh} &= \texttt{EltMesh object} \\
\texttt{type} : \texttt{simplex} \\
\texttt{order} : 1 \\
\texttt{d} : 2 \\
\texttt{n} : 2 \\
\texttt{q} : (2,1002001) \\
\texttt{me} : (3,2000000)
\end{align*}
\]

\[
\begin{align*}
\texttt{Eh} &= \texttt{fc\_hypermesh.EltMesh}(m,q,me, key,value,\ldots)
\end{align*}
\]

Some optional \texttt{key/value} pairs arguments are available with \texttt{key}:

• \texttt{’type’} : used to select the kind of elements used. The default value is \texttt{’simplex’} and otherwise \texttt{’orthotope’} can be used.

• \texttt{’order’} : gives the order of the mesh elements (default is 1).
Listing 23: EltMesh constructor in dimension $d=3$ (orthotope mesh)

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

<table>
<thead>
<tr>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eh = EltMesh object</td>
</tr>
<tr>
<td>type : orthotope</td>
</tr>
<tr>
<td>order : 2</td>
</tr>
<tr>
<td>$d$ : 3</td>
</tr>
<tr>
<td>$m$ : 3</td>
</tr>
<tr>
<td>$q$ : (3,693)</td>
</tr>
<tr>
<td>$me$ : (27,60)</td>
</tr>
</tbody>
</table>

### 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**

```matlab
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;

37
with \( obj.d=1 \) and \( obj.m=0 \), \texttt{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: \texttt{plotmesh} method of a 2D \texttt{EltMesh} objects, \texttt{figure(1)} (left) and \texttt{figure(2)} (right)

### B.1.3 \texttt{plotnodes} method

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

**Syntaxe**

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

**Description**

\texttt{obj.plotnodes()}

Uses \texttt{scatter} function (1D and 2D) and \texttt{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.

\texttt{obj.plotnodes(key, value, ...)}

Some optional \texttt{key/value} pairs arguments are available with \texttt{key}:

- '\texttt{vcolor}' : use to specify the point color for the mesh vertices. Default is \texttt{obj.color}.
- '\texttt{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.

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.

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**

```matlab
obj.ploteltsNumber()
```

Uses `fc_hyperMesh.plotElementsNumber` function to represent node numbers.

```matlab
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 between vertices and barycenter of the mesh elements. Default is `'` (no lines).
- `'vLineStyle'` : Select lines type. Default is `'-'` (dotted lines).
- `'vLineWidth'` : Set lines width. Default is 0.5.

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

```matlab
[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
```

```matlab
[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)
References