fc-hypermesh Python package, User’s Guide * version 0.1.1
Francois Cuvelier

To cite this version:
Francois Cuvelier. fc-hypermesh Python package, User’s Guide * version 0.1.1. 2019. hal-02425548

HAL Id: hal-02425548
https://hal-univ-paris13.archives-ouvertes.fr/hal-02425548
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 Python package allows in any dimension $d$ to generate conforming meshes of $d$-orthotopes by $p$-order simplices or orthotopes with their $m$-faces. It was created to show the implementation of the algorithms of [1]. The package uses Python objects and is provided with meshes visualization tools for dimension less than or equal to 3.

0 Contents

1 Introduction 2
2 Installation and uninstall 6
3 Classes of the package 7
3.1 Class EltMesh 7
3.2 Class OrthMesh 7
3.2.1 Constructor 8
3.2.2 Access to OrthMesh’s fields 10
3.2.3 plotmesh method 11
3.2.4 plotnodes method 13
3.2.5 ploteltsNumber method 14
3.2.6 ploteltsNumber method 15
4 Using the fc_hypermesh package 16
4.1 2d-orthotope meshing by simplices 17
4.2 3d-orthotope meshing by simplices 18
4.3 2d-orthotope meshing by orthotopes 19
4.4 3d-orthotope meshing by orthotopes 20
4.5 Mapping of a 2d-orthotope meshing by simplices 21
4.6 Mapping of a 3d-orthotope meshing by orthotopes 22
1 Introduction

The \texttt{hypemesh} package contains a simple class \texttt{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

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

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>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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><img src="image1.png" alt="Diagram 1" /></td>
<td><img src="image2.png" alt="Diagram 2" /></td>
<td><img src="image3.png" alt="Diagram 3" /></td>
<td><img src="image4.png" alt="Diagram 4" /></td>
</tr>
<tr>
<td>2</td>
<td><img src="image5.png" alt="Diagram 5" /></td>
<td><img src="image6.png" alt="Diagram 6" /></td>
<td><img src="image7.png" alt="Diagram 7" /></td>
<td><img src="image8.png" alt="Diagram 8" /></td>
</tr>
<tr>
<td>3</td>
<td><img src="image9.png" alt="Diagram 9" /></td>
<td><img src="image10.png" alt="Diagram 10" /></td>
<td><img src="image11.png" alt="Diagram 11" /></td>
<td><img src="image12.png" alt="Diagram 12" /></td>
</tr>
</tbody>
</table>

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

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 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: 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.
2 Installation and uninstall

This toolbox was tested on various OS with Python releases (from python.org):

<table>
<thead>
<tr>
<th>OS</th>
<th>Python</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2.7.16</td>
</tr>
<tr>
<td></td>
<td>3.5.9</td>
</tr>
<tr>
<td></td>
<td>3.6.10</td>
</tr>
<tr>
<td></td>
<td>3.7.6</td>
</tr>
<tr>
<td></td>
<td>3.8.1</td>
</tr>
<tr>
<td>CentOS 7.7.1908</td>
<td>✓</td>
</tr>
<tr>
<td>Debian 9.11</td>
<td>✓</td>
</tr>
<tr>
<td>Fedora 29</td>
<td>✓</td>
</tr>
<tr>
<td>OpenSUSE Leap 15.0</td>
<td>✓</td>
</tr>
<tr>
<td>Ubuntu 18.04.3 LTS</td>
<td>✓</td>
</tr>
<tr>
<td>Apple Mac OS X</td>
<td>2.7.16</td>
</tr>
<tr>
<td>MacOS High Sierra 10.13.6</td>
<td>✓</td>
</tr>
<tr>
<td>MacOS Mojave 10.14.4</td>
<td>✓</td>
</tr>
<tr>
<td>MacOS Catalina 10.15.2</td>
<td>✓</td>
</tr>
<tr>
<td>Microsoft Windows</td>
<td>2.7.16</td>
</tr>
<tr>
<td>Windows 10 (1909)</td>
<td>✓</td>
</tr>
</tbody>
</table>

**Installation:**

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

  ```bash
  $ pip install -U --user fc_hypermesh
  ```

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

  ```bash
  $ sudo pip install -U fc_hypermesh
  ```

Another way is to download the required archive and to make the installation from the downloaded file.

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

  ```bash
  $ pip install <PATH_TO_FOLDER>/fc_hypermesh-<VERSION>.tar.gz --user -U
  ```

  where <PATH_TO_FOLDER> will be replaced by the path to the saved archive and <VERSION> by the version of the archive.

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

  ```bash
  $ sudo pip install <PATH_TO_FOLDER>/fc_hypermesh-<VERSION>.tar.gz -U
  ```
Uninstall: To uninstall this package, you only have to execute one of these commands depending on the type of installation performed.

$ pip uninstall fc_hypermesh

or

$ sudo pip uninstall fc_hypermesh

3 Classes of the package

First of all, the low level class EltMesh is presented. Thereafter the main class OrthMesh, which is an union of EltMesh objects, is described.

3.1 Class EltMesh

An elementary mesh class EltMesh is used to store only one mesh in space dimension $d$, the main mesh as well as any of the meshes of the $m$-faces. This elementary mesh is made either with $p$-order simplices or with $p$-order orthotopes. This class EltMesh also simplify (for me) the codes writing. Its attributes are the following:

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

3.2 Class OrthMesh

The aim of the class OrthMesh is to efficiently create an object which contains a main mesh of a $d$-orthotope and all its $m$-face meshes deduced from the main mesh. All meshes are made either with $p$-order simplices or with $p$-order orthotopes.

Let the $d$-orthotope defined by $[a_1, b_1] \times \cdots \times [a_d, b_d]$. The class OrthMesh corresponding to this $d$-orthotope contains the main mesh and all the meshes of its $m$-faces, $0 \leq m < d$. Its attributes are the following:

- $d$: space dimension.
- $type$: string 'simplex' or 'orthotope' mesh.
- $order$: order $p$ of the elements.
- $Mesh$: main mesh as an EltMesh object.
- $Faces$: 2d-list of EltMesh objects such that $Faces[0]$ is a list of all the meshes of the $(d-1)$-faces, $Faces[1]$ is a list of all the meshes of the $(d-2)$-faces, and so on
- $box$: a $d$-by-$2$ numpy array such that $box[i-1][0]$ is $a_i$ value and $box[i-1][1]$ is $b_i$ value.
In all the Python examples given in the following section, we suppose the following code previously load:

Listing 1: Previously load code for Python examples

```python
import fc_hypermesh
import matplotlib.pyplot as plt
from fc_tools.Matplotlib import set_axes_equal
```

3.2.1 Constructor

The OrthMesh constructor is:

### Syntaxe

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

### Description

**Oh = OrthMesh(d,N)**

Builds an OrthMesh object where N is either a 1-by-d array/list such that N[i−1] is the number of discretization for [a_i, b_i] = [0, 1] or either an integer if the the number of discretization is the same in all space directions. By default, the output OrthMesh object is made with 1-order simplices.

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

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

- **box**: where value is a d-by-2 list or numpy array such that value[i−1][0] is a_i value and value[i−1][1] is b_i value. Default is [0, 1]^d.
- **type**: The default value for optional key parameter type is 'simplex' and otherwise 'orthotope' can be used.

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

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

```
Output
```OrthMesh object
d : 3
order : 1
box : [[0.0, 1.0], [0.0, 1.0], [0.0, 1.0]]
mapping : None
Mesh (order,type,nq,nme) : (1,orthotope,1331,1000)
Number of 2-faces : 6
 [ 0] (order,type,nq,nme) : (1,orthotope,121,100)
 [ 1] (order,type,nq,nme) : (1,orthotope,121,100)
 [ 2] (order,type,nq,nme) : (1,orthotope,121,100)
 [ 3] (order,type,nq,nme) : (1,orthotope,121,100)
 [ 4] (order,type,nq,nme) : (1,orthotope,121,100)
 [ 5] (order,type,nq,nme) : (1,orthotope,121,100)
Number of 1-faces : 12
 [ 0] (order,type,nq,nme) : (1,orthotope,11,10)
 [ 1] (order,type,nq,nme) : (1,orthotope,11,10)
 [ 2] (order,type,nq,nme) : (1,orthotope,11,10)
 [ 3] (order,type,nq,nme) : (1,orthotope,11,10)
 [ 4] (order,type,nq,nme) : (1,orthotope,11,10)
 [ 5] (order,type,nq,nme) : (1,orthotope,11,10)
 [ 6] (order,type,nq,nme) : (1,orthotope,11,10)
 [ 7] (order,type,nq,nme) : (1,orthotope,11,10)
 [ 8] (order,type,nq,nme) : (1,orthotope,11,10)
 [ 9] (order,type,nq,nme) : (1,orthotope,11,10)
[10] (order,type,nq,nme) : (1,orthotope,11,10)
[11] (order,type,nq,nme) : (1,orthotope,11,10)
Number of 0-faces : 8
 [ 0] (order,type,nq,nme) : (1,orthotope,1)
 [ 1] (order,type,nq,nme) : (1,orthotope,1)
 [ 2] (order,type,nq,nme) : (1,orthotope,1)
 [ 3] (order,type,nq,nme) : (1,orthotope,1)
 [ 4] (order,type,nq,nme) : (1,orthotope,1)
 [ 5] (order,type,nq,nme) : (1,orthotope,1)
 [ 6] (order,type,nq,nme) : (1,orthotope,1)
 [ 7] (order,type,nq,nme) : (1,orthotope,1)
```

- **order = value**: gives the order of the mesh elements (default is 1).
Listing 3: OrthMesh constructor in dimension d=3 (simplicial mesh)

```
Oh = fc_hypermesh.OrthMesh(2,[10,20],order=4)
print(Oh)
```

Output

```
OrthMesh object
d : 2
order : 4
box : [[0.0, 1.0], [0.0, 1.0]]
mapping : None
Mesh (order,type,nq,nme) : (4,simplex,3321,400)
Number of 1-faces : 4
[ 0] (order,type,nq,nme) : (4,simplex,81,20)
[ 1] (order,type,nq,nme) : (4,simplex,81,20)
[ 2] (order,type,nq,nme) : (4,simplex,41,10)
[ 3] (order,type,nq,nme) : (4,simplex,41,10)
Number of 0-faces : 4
[ 0] (order,type,nq,nme) : (1,simplex,1,1)
[ 1] (order,type,nq,nme) : (1,simplex,1,1)
[ 2] (order,type,nq,nme) : (1,simplex,1,1)
[ 3] (order,type,nq,nme) : (1,simplex,1,1)
```

• `box = value`: used to specify the d-orthotope $[a_1, b_1] \times \cdots \times [a_d, b_d]$ by setting value as a d-by-2 array such that $a_i = \text{value}(i,1)$ and $b_i = \text{value}(i,2)$.

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

```
Oh = fc_hypermesh.OrthMesh(2,10, box=[[-1,1],[-2,2]])
print('Oh/uni2423=
'+str(Oh))
```

Output

```
Oh = OrthMesh object
d : 2
order : 1
box : [[-1.0, 1.0], [-2.0, 2.0]]
mapping : None
Mesh (order,type,nq,nme) : (1,simplex,121,200)
Number of 1-faces : 4
[ 0] (order,type,nq,nme) : (1,simplex,11,10)
[ 1] (order,type,nq,nme) : (1,simplex,11,10)
[ 2] (order,type,nq,nme) : (1,simplex,11,10)
[ 3] (order,type,nq,nme) : (1,simplex,11,10)
Number of 0-faces : 4
[ 0] (order,type,nq,nme) : (1,simplex,1,1)
[ 1] (order,type,nq,nme) : (1,simplex,1,1)
[ 2] (order,type,nq,nme) : (1,simplex,1,1)
[ 3] (order,type,nq,nme) : (1,simplex,1,1)
```

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

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

```
Oh = fc_hypermesh.OrthMesh(3,10, m_min=2)
print('Oh/uni2423=
'+str(Oh))
print('q/uni2423=
'+str(Oh.Mesh.q))
```

Output

```
Oh = OrthMesh object
d : 3
order : 1
box : [[0.0, 1.0], [0.0, 1.0], [0.0, 1.0]]
mapping : None
Mesh (order,type,nq,nme) : (1,simplex,1331,6000)
Number of 2-faces : 6
[ 0] (order,type,nq,nme) : (1,simplex,121,200)
[ 1] (order,type,nq,nme) : (1,simplex,121,200)
[ 2] (order,type,nq,nme) : (1,simplex,121,200)
[ 3] (order,type,nq,nme) : (1,simplex,121,200)
[ 4] (order,type,nq,nme) : (1,simplex,121,200)
[ 5] (order,type,nq,nme) : (1,simplex,121,200)
q =
[[0. 0.1 0.2 ... 0.8 0.9 1. ]
[0. 0. 0. ... 1. 1. 1. ]
[0. 0. 0. ... 1. 1. 1. ]]
```

• `mapping=value`: used to apply on the mesh a mapping function given by a function handle.
import numpy as np

mfun = lambda q: np.array([q[0]+np.sin(q[1]),q[1],q[2]])

Oh = fc_hypermesh.OrthMesh(3,10, _m_min=2, mapping=mfun)
print('Oh/uni2423=
' + str(Oh))

print('q/uni2423=
' + str(Oh.Mesh.q))

Output

Oh =
OrthMesh object
d : 3
order : 1
box : [[0.0, 1.8414709848078965], [0.0, 1.0], [0.0, 1.0]]
mapping : lambda q: np.array([q[0]+np.sin(q[1]),q[1],q[2]])

Mesh(order,type,mq,me) : (1,simplex,1331,6000)
Number of 2-faces : 6
[ 0] (order,type,mq,me) : (1,simplex,1331,6000)
[ 1] (order,type,mq,me) : (1,simplex,1331,6000)
[ 2] (order,type,mq,me) : (1,simplex,1331,6000)
[ 3] (order,type,mq,me) : (1,simplex,1331,6000)
[ 4] (order,type,mq,me) : (1,simplex,1331,6000)
[ 5] (order,type,mq,me) : (1,simplex,1331,6000)

q =
[[0.  0.1  0.2  ...  1.64147098  1.74147098  1.84147098]
 [0.  0.  0.    ...  1.  1.  1.    1.  1.  1.  1.]]

3.2.2 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)
print(Oh)

Output

Oh =
OrthMesh object
d : 3
order : 2
box : [[0.0, 1.0], [0.0, 1.0], [0.0, 1.0]]
mapping : None

Mesh(order,type,mq,me) : (2,simplex,9261,6000)
Number of 2-faces : 6
[ 0] (order,type,mq,me) : (2,simplex,441,200)
[ 1] (order,type,mq,me) : (2,simplex,441,200)
[ 2] (order,type,mq,me) : (2,simplex,441,200)
[ 3] (order,type,mq,me) : (2,simplex,441,200)
[ 4] (order,type,mq,me) : (2,simplex,441,200)
[ 5] (order,type,mq,me) : (2,simplex,441,200)

Number of 1-faces : 12
[ 0] (order,type,mq,me) : (2,simplex,21,10)
[ 1] (order,type,mq,me) : (2,simplex,21,10)
[ 2] (order,type,mq,me) : (2,simplex,21,10)
[ 3] (order,type,mq,me) : (2,simplex,21,10)
[ 4] (order,type,mq,me) : (2,simplex,21,10)
[ 5] (order,type,mq,me) : (2,simplex,21,10)
[ 6] (order,type,mq,me) : (2,simplex,21,10)
[ 7] (order,type,mq,me) : (2,simplex,21,10)
[ 8] (order,type,mq,me) : (2,simplex,21,10)
[ 9] (order,type,mq,me) : (2,simplex,21,10)
[10] (order,type,mq,me) : (2,simplex,21,10)
[11] (order,type,mq,me) : (2,simplex,21,10)

Number of 0-faces : 8
[ 0] (order,type,mq,me) : (1,simplex,1,1)
[ 1] (order,type,mq,me) : (1,simplex,1,1)
[ 2] (order,type,mq,me) : (1,simplex,1,1)
[ 3] (order,type,mq,me) : (1,simplex,1,1)
[ 4] (order,type,mq,me) : (1,simplex,1,1)
[ 5] (order,type,mq,me) : (1,simplex,1,1)
[ 6] (order,type,mq,me) : (1,simplex,1,1)
[ 7] (order,type,mq,me) : (1,simplex,1,1)

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
print('Th/uni2423=
' + str(Th))

Output

Th = EltMesh object
type : 0 (simplex)
order : 2
labels : 1
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 $Oh\cdot Faces[Oh\cdot d-m-1][k-1]$.

```
Fh=Oh\cdot Faces[0][2]
print('Fh/uni2423=/uni2423'+ str(Fh))
```

```
Fh = EltMesh object  
  type : 0 (simplex)  
  order : 2  
  label : 3  
  m : 2  
  d : 3  
  q : (3,441)  
  me : (6,200)
```

One can easily access to each field of an EltMesh object (see section [3.1]). For example, to access the nodes array and the connectivity array of the $Th$ EltMesh object we do respectively $Th\cdot q$ and $Th\cdot me$. We also have the following link between global nodes array $Th\cdot q$ and local nodes array $Fh\cdot q$:

$$Th\cdot q[:,Fh\cdot toGlobal]==Fh\cdot q$$

or more generally, for all $m$ in $[0,Oh\cdot d]$ and for all $k$ in $[1,E^d_m]$:

$$Oh\cdot Mesh\cdot q[:,Oh\cdot Faces[Oh\cdot d-m-1][k-1].toGlobal]==Oh\cdot Faces[Oh\cdot d-m-1][k-1].q$$

where $E^d_m$ is defined in [1].

### 3.2.3 plotmesh method

The `plotmesh()` member function uses the Matplotlib Python package [2, 3] to represent the mesh given by an OrthMesh object.

**Syntax**

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

**Description**

- `obj.plotmesh()` plot the main mesh.

- `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. ($0 \leq m \leq d$)
  - `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 Matplotlib function used:

- with `obj.d=3` and `obj.m=3`, `Line3DCollection` function is used;
- with `obj.d=3` and `obj.m=2`, `Poly3DCollection` or `Line3DCollection` function are used;
- with `obj.d=3` and `obj.m=1`, `Line3DCollection` function is used;
- with `obj.d=3` and `obj.m=0`, `scatter` function is used;
- with `obj.d=2` and `obj.m=2`, `PolyCollection` function is used;
- with `obj.d=2` and `obj.m=1`, `plot` function is used;
- with `obj.d=2` and `obj.m=0`, `scatter` function is used;
- with `obj.d=1` and `obj.m=1`, `plot` function is used;
- with `obj.d=1` and `obj.m=0`, `scatter` function is used;
**Examples**  In Listing 7 and Listing 8, the code given in Listing 4 is supposed to be preloaded.

```python
Oh1=fc_hypermesh.OrthMesh(3,6,box=[[-1,1],[-2,2],[0,3]])
plt.figure(1)
Oh1.plotmesh()
set_axes_equal()
Oh2=fc_hypermesh.OrthMesh(3,6,type='orthotope')
plt.figure(2)
Oh2.plotmesh(color='Salmon', linewidth=2)
set_axes_equal()
```

**Listing 7:** `plotmesh` method of 3D OrthMesh objects tessellated with simplices in `figure(1)` (left) and with orthotopes in `figure(2)` (right)
3.2.4 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

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

Description

- `obj.plotnodes()`
  - Uses Matplotlib scatter function 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. \((0 \leq m \leq d)\)
• 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 function.

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

Description

```
obj.plotnodesNumber()
```
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 nodes 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 `'auto'` to automatically set to the color of the main mesh (if `m = d`) or to the colors of the `m`-faces currently drawn.
- `ec`: use to specify the color of box outline. Default is `'w'` (i.e. white). Value `'auto'` can be used.
- `fc`: use to specify text background color. Default is `'w'` (i.e. white). Value `'auto'` can be used.
- `vLineColor`: Draw lines with value as color between vertices and barycenter of the mesh elements. Default is `None` (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 Matplotlib `text` function in 3D or those of the Matplotlib `annotate` function in 2D.

```
Oh=fc_hypermesh.OrthMesh(2,4,order=3)
plt.figure(1)
Oh.plotmesh(color='LightGray')
Oh.ploteltsNumber()
set_axes_equal();plt.axis('off')
plt.figure(2)
Oh.plotmesh()
Oh.ploteltsNumber(color='r',ec='r', fontsize=7, vLineStyle='auto', fc='w')
set_axes_equal();plt.axis('off')
```

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

3.2.6 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.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 ≤ `m` ≤ `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 `'auto'` to automatically set to the color of the main mesh (if `m = d`) or to the colors of the `m`-faces currently drawn.
- **ec**: use to specify the color of box outline. Default is `'w'` (i.e. white). Value `'auto'` can be used.
- **fc**: use to specify text background color. Default is `'w'` (i.e. white). Value `'auto'` can be used.
- **vLineColor**: Draw lines with `value` as color between vertices and barycenter of the mesh elements. Default is `None` (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 Matplotlib `text` function in 3D or those of the Matplotlib `annotate` function in 2D.

```python
Oh=fc_hypermesh.OrthMesh(2,4,order=3)
plt.figure(1)
Oh.plotmesh(color='LightGray')
Oh.ploteltsNumber()
set_axes_equal();plt.axis('off')
plt.figure(2)
Oh.plotmesh()
Oh.ploteltsNumber(color='r',ec='r', fontsize=7, vLineColor='auto', fc='w')
set_axes_equal();plt.axis('off')
```

Listing 11: `ploteltsNumber` method of a 2D `EltMesh` objects, `figure(1)` (left) and `figure(2)` (right)

4 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 48.096 GB in memory. With 3-order simplices, the OrthMesh object uses 617.764 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 Section 5 for more details.

<table>
<thead>
<tr>
<th>d</th>
<th>OrthMesh (orthotopes)</th>
<th>OrthMesh (simplices)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>296 B</td>
<td>296 B</td>
</tr>
<tr>
<td>2</td>
<td>6 KB</td>
<td>8 KB</td>
</tr>
<tr>
<td>3</td>
<td>144 KB</td>
<td>282 KB</td>
</tr>
<tr>
<td>4</td>
<td>2 MB</td>
<td>12 MB</td>
</tr>
<tr>
<td>5</td>
<td>57 MB</td>
<td>696 MB</td>
</tr>
<tr>
<td>6</td>
<td>1.163 GB</td>
<td>48.096 GB</td>
</tr>
<tr>
<td>7</td>
<td>23.815 GB</td>
<td>3.845 TB</td>
</tr>
<tr>
<td>8</td>
<td>496.025 GB</td>
<td>346.756 TB</td>
</tr>
</tbody>
</table>

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>616 B</td>
<td>616 B</td>
</tr>
<tr>
<td>2</td>
<td>32 KB</td>
<td>35 KB</td>
</tr>
<tr>
<td>3</td>
<td>1 MB</td>
<td>1 MB</td>
</tr>
<tr>
<td>4</td>
<td>64 MB</td>
<td>115 MB</td>
</tr>
<tr>
<td>5</td>
<td>2.695 GB</td>
<td>7.737 GB</td>
</tr>
<tr>
<td>6</td>
<td>109.134 GB</td>
<td>617.764 GB</td>
</tr>
<tr>
<td>7</td>
<td>4.352 TB</td>
<td>58.136 TB</td>
</tr>
<tr>
<td>8</td>
<td>171.900 TB</td>
<td>6247.498 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 \( N = 10 \).

In all the next examples, the following code is previously loaded:

```python
import matplotlib.pyplot as plt
from fc_tools.colors import str2rgb
from fc_hypermesh import OrthMesh
from fc_tools.Matplotlib import DisplayFigures, set_axes_equal
```

### 4.1 2d-orthotope meshing by simplices

In Listing 12 an OrthMesh object is built under Python for the orthotope \([-1, 1] \times [0, 1]\) with simplicial elements and \( N = (12, 5) \). The main mesh and all the \( m \)-face meshes of the resulting object are plotted.
In Listing 12, an OrthMesh object is built under Python 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.

4.2 3d-orthotope meshing by simplices
In Listing 13, an OrthMesh object is built under Python for the orthotope $[-1, 1] \times [0, 1]$ with orthotope elements and $N = (10, 5, 10)$. The main mesh and all the $m$-face meshes of the resulting object are plotted.
4.4 3d-orthotope meshing by orthotopes

In Listing 14 an OrthMesh object is built under Python 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 $m$-face meshes of the resulting object are plotted.
Listing 15: 3D orthotope \texttt{OrthMesh} object with Python 3.8.1, main mesh (upper left), 2-face meshes (upper right), 1-face meshes (bottom left) and 0-face meshes (bottom right).

4.5 Mapping of a 2d-orthotope meshing by simplices

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

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

$$ \begin{pmatrix} x \\ y \end{pmatrix} \rightarrow F(x, y) = \begin{pmatrix} 20x \\ 2(2y - 1 + \cos(2\pi x)) \end{pmatrix} $$

21
import numpy as np
trans=lambda q: np.array([20*q[0],2*(2*q[1]-1+np.cos(2*np.pi*q[0])))])
oTh=OrthMesh(2,[100,20],type='simplex',mapping=trans)
plt.figure(1)
oTh.plotmesh(legend=True)
plt.axis('equal')
plt.figure(2)
oTh.plotmesh(color='lightgray')
oTh.plotmesh(m=1,legend=True,linewidth=3)
plt.axis('equal')
plt.axis('off')
plt.figure(3)
oTh.plotmesh(color='lightgray')
oTh.plotmesh(m=1,color='black')
oTh.plotmesh(m=0,legend=True,s=105)
plt.axis('equal')
plt.axis('off')

Listing 16: Mapping of a 2D simplicial OrthMesh object with Python 3.8.1, main mesh (upper left), 1-face meshes (upper right), and 0-face meshes (bottom)

4.6  Mapping of a 3d-orthotope meshing by orthotopes

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

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

\[
\begin{pmatrix}
x \\
y \\
z
\end{pmatrix} \longrightarrow F(x,y,z) = \begin{pmatrix}
x + \sin(4\pi y) \\
10y \\
z + \cos(4\pi y)
\end{pmatrix}
\]
import numpy as np
trans=lambda q: np.array([q[0]+np.sin(4*np.pi*q[1]), 10*q[1]-1, q[2]+np.cos(4*np.pi*q[1])])
oTh=OrthMesh(3, [3, 25, 3], type='simplex', mapping=trans)
plt.figure(1)
oTh.plotmesh(legend=True)
set_axes_equal()
plt.figure(2)
oTh.plotmesh(m=2, legend=True, edgecolor=[0,0,0])
set_axes_equal()
plt.figure(3)
oTh.plotmesh(m=2, edgecolor='lightgray', facecolor=None, alpha=0.3)
set_axes_equal()
oTh.plotmesh(m=1, legend=True, linewidth=2)
set_axes_equal()
plt.figure(4)
oTh.plotmesh(m=1, color='black')
oTh.plotmesh(m=0, legend=True, s=55)
set_axes_equal()
Listing 17: Mapping of a 3D orthotope OrthMesh object with Python 3.8.1, main mesh (upper left), 2-face meshes (upper right), 1-face meshes (bottom left) and 0-face meshes (bottom right)

5 Memory consuming

Take care when using these codes with memory consuming: the number of points $n_q$ and the number of elements increases exponentially according to the space dimension $d$. If $(N+1)$ points are taken in each space direction, we have

$$n_q = (pN + 1)^d,$$

for both tessellation and triangulation

and

$$n_{me} = \begin{cases} N^d, & \text{for tessellation by orthotopes} \\ d!N^d, & \text{for tessellation by simplices.} \end{cases}$$
If the array \( q \) is stored as double (8 octets) then

\[
\text{mem. size of } q = d \times n_q \times 8 \text{ octets}
\]

and if the array \( me \) as int (8 octets) then

\[
\text{mem. size of } me = \begin{cases} 2^d \times n_{me} \times 8 \text{ octets} & \text{(tessellation by orthotopes)} \\ (d + 1) \times n_{me} \times 8 \text{ octets} & \text{(tessellation by simplices)} \end{cases}
\]

For \( N = 10 \) and \( d \in [1, 8] \), 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.

### 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>
<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>11</td>
<td>100</td>
<td>200</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>10 000</td>
<td>240 000</td>
</tr>
<tr>
<td>4</td>
<td>11</td>
<td>100 000</td>
<td>12 000 000</td>
</tr>
<tr>
<td>5</td>
<td>11</td>
<td>1 000 000</td>
<td>720 000 000</td>
</tr>
<tr>
<td>6</td>
<td>11</td>
<td>10 000 000</td>
<td>50 400 000 000</td>
</tr>
<tr>
<td>7</td>
<td>11</td>
<td>100 000 000</td>
<td>4 032 000 000 000</td>
</tr>
<tr>
<td>8</td>
<td>11</td>
<td>1 000 000 000</td>
<td>40 944 000 000 000</td>
</tr>
</tbody>
</table>

### Table 6: Memory usage of the array \( q \) and the array \( me \) for the tessellation of an orthotope by orthotopes and by simplices according to the space dimension \( d \) and with \( N = 10 \).

<table>
<thead>
<tr>
<th>( d )</th>
<th>( q ) me (orthotopes)</th>
<th>( me ) (simplices)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>88 o</td>
<td>80 o</td>
</tr>
<tr>
<td>2</td>
<td>176 o</td>
<td>1 ko</td>
</tr>
<tr>
<td>3</td>
<td>264 o</td>
<td>32 ko</td>
</tr>
<tr>
<td>4</td>
<td>352 o</td>
<td>640 ko</td>
</tr>
<tr>
<td>5</td>
<td>440 o</td>
<td>12 Mo</td>
</tr>
<tr>
<td>6</td>
<td>528 o</td>
<td>256 Mo</td>
</tr>
<tr>
<td>7</td>
<td>616 o</td>
<td>5 Go</td>
</tr>
<tr>
<td>8</td>
<td>704 o</td>
<td>102 Go</td>
</tr>
</tbody>
</table>

---

### 6 Benchmarks

For all the following tables, the computational costs of the OrthMesh constructor are given for the orthotope \([-1, 1]^d\) under Python 3.8.1. The computations were done on a laptop with Intel(R) Core(TM) i9-7940X CPU @ 3.10GHz processor and 63Go of RAM under Ubuntu 18.04.3 LTS (x86_64).

In the following pages, computational costs of the OrthMesh constructor will be done by using bench01 function. As sample, we give an example with output. Thereafter, all the output will be presented in tabular form.
```python
from fc_hypermesh import bench
bench(3, range(20, 170, 20), type='simplex', box=[[-1, 1], [-1, 1], [-1, 1]], order=1)
```

Listing 18: bench sample

```
# Output

# computer: cosmos-ubuntu-18-04
# system: Ubuntu 18.04.3 LTS (x86_64)
# processor: Intel(R) Core(TM) i9-7940X CPU @ 3.10GHz
# (1 procs/14 cores by proc/2 threads by core)
# RAM: 62.6 Go
# software: Python
# release: 3.8.1

# fc_hypermesh.OrthMesh constructor with
d = 3
# type = simplex
# order = 1
# box = [-1, 1], [-1, 1], [-1, 1]
# mapping=None

# date: 2019-12-30_09-33-14
# nbruns: 5
# numpy: i8 i8 i8 f8
# format: {:>7d} {:>10d} {:>10d} {:11.3f}
# labels: N nq nme OrthMesh(s)
20 9261 48000 0.185
40 68921 384000 0.190
60 226981 1296000 0.268
80 531441 3072000 0.346
100 1030301 6000000 0.472
120 1771561 10368000 0.686
140 2803221 16464000 0.983
160 4173281 24576000 1.328
```

6.1 Tessellation by orthotopes

```python
from fc_hypermesh import bench
bench(2, range(1000, 6000, 1000), type='orthotope', box=[[-1, 1], [-1, 1]], order=1)
```

Listing 19: Tessellation of $[-1, 1]^2$ by orthotopes

<table>
<thead>
<tr>
<th>$N$</th>
<th>$n_q$</th>
<th>$n_{me}$</th>
<th>Python</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>1   002 001</td>
<td>1   000 000</td>
<td>0.129</td>
</tr>
<tr>
<td>2000</td>
<td>4   004 001</td>
<td>4   000 000</td>
<td>0.327</td>
</tr>
<tr>
<td>3000</td>
<td>9   006 001</td>
<td>9   000 000</td>
<td>0.666</td>
</tr>
<tr>
<td>4000</td>
<td>16  008 001</td>
<td>16  000 000</td>
<td>1.12</td>
</tr>
<tr>
<td>5000</td>
<td>25  010 001</td>
<td>25  000 000</td>
<td>1.729</td>
</tr>
</tbody>
</table>

Table 7: Tessellation of $[-1, 1]^2$ by orthotopes

```python
from fc_hypermesh import bench
bench(3, range(50, 400, 50), type='orthotope', box=[[-1, 1], [-1, 1], [-1, 1]], order=1)
```

Listing 20: Tessellation of $[-1, 1]^3$ by orthotopes
Table 8: Tessellation of $[-1,1]^3$ by orthotopes

<table>
<thead>
<tr>
<th>$N$</th>
<th>$n_q$</th>
<th>$n_{mc}$</th>
<th>Python</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>132 651</td>
<td>125 000</td>
<td>0.139</td>
</tr>
<tr>
<td>100</td>
<td>1 030 301</td>
<td>1 000 000</td>
<td>0.269</td>
</tr>
<tr>
<td>150</td>
<td>3 442 951</td>
<td>3 375 000</td>
<td>0.536</td>
</tr>
<tr>
<td>200</td>
<td>8 120 601</td>
<td>8 000 000</td>
<td>1.058</td>
</tr>
<tr>
<td>250</td>
<td>15 813 251</td>
<td>15 625 000</td>
<td>1.89</td>
</tr>
<tr>
<td>300</td>
<td>27 270 901</td>
<td>27 000 000</td>
<td>3.132</td>
</tr>
<tr>
<td>350</td>
<td>43 243 551</td>
<td>42 875 000</td>
<td>4.824</td>
</tr>
</tbody>
</table>

Listing 21: Tessellation of $[-1,1]^4$ by orthotopes

from fc_hypermesh import bench
bench(4,[10,20,30,40,50,62],type='orthotope',order=1,
      box=[[-1,1],[-1,1],[-1,1],[-1,1]])

Table 9: Tessellation of $[-1,1]^4$ by orthotopes

<table>
<thead>
<tr>
<th>$N$</th>
<th>$n_q$</th>
<th>$n_{mc}$</th>
<th>Python</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>16 441</td>
<td>10 000</td>
<td>0.206</td>
</tr>
<tr>
<td>20</td>
<td>194 481</td>
<td>160 000</td>
<td>0.294</td>
</tr>
<tr>
<td>30</td>
<td>923 521</td>
<td>810 000</td>
<td>0.399</td>
</tr>
<tr>
<td>40</td>
<td>2 825 761</td>
<td>2 560 000</td>
<td>0.717</td>
</tr>
<tr>
<td>50</td>
<td>6 765 201</td>
<td>6 250 000</td>
<td>1.488</td>
</tr>
<tr>
<td>62</td>
<td>15 752 961</td>
<td>14 776 336</td>
<td>3.053</td>
</tr>
</tbody>
</table>

Listing 22: Tessellation of $[-1,1]^5$ by orthotopes

from fc_hypermesh import bench
bench(5,[5,10,15,20,25,27],type='orthotope',order=1,
      box=[[-1,1],[-1,1],[-1,1],[-1,1],[-1,1]])

Table 10: Tessellation of $[-1,1]^5$ by orthotopes

<table>
<thead>
<tr>
<th>$N$</th>
<th>$n_q$</th>
<th>$n_{mc}$</th>
<th>Python</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>7 776</td>
<td>3 125</td>
<td>0.383</td>
</tr>
<tr>
<td>10</td>
<td>161 051</td>
<td>100 000</td>
<td>0.467</td>
</tr>
<tr>
<td>15</td>
<td>1 048 576</td>
<td>759 375</td>
<td>0.751</td>
</tr>
<tr>
<td>20</td>
<td>4 084 101</td>
<td>3 200 000</td>
<td>1.562</td>
</tr>
<tr>
<td>25</td>
<td>11 881 376</td>
<td>9 765 625</td>
<td>3.955</td>
</tr>
<tr>
<td>27</td>
<td>17 210 368</td>
<td>14 348 907</td>
<td>5.535</td>
</tr>
</tbody>
</table>

6.2 Tessellation by simplices

from fc_hypermesh import bench
bench(2,range(1000,6000,1000),type='simplex',box=[[-1,1],[-1,1]],order=1)

Listing 23: Tessellation of $[-1,1]^2$ by simplices

Table 11: Tessellation of $[-1,1]^2$ by simplices

<table>
<thead>
<tr>
<th>$N$</th>
<th>$n_q$</th>
<th>$n_{mc}$</th>
<th>Python</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>1 002 001</td>
<td>2 000 000</td>
<td>0.199</td>
</tr>
<tr>
<td>2000</td>
<td>4 004 001</td>
<td>8 000 000</td>
<td>0.473</td>
</tr>
<tr>
<td>3000</td>
<td>9 006 001</td>
<td>18 000 000</td>
<td>0.916</td>
</tr>
<tr>
<td>4000</td>
<td>16 008 001</td>
<td>32 000 000</td>
<td>1.528</td>
</tr>
<tr>
<td>5000</td>
<td>25 010 001</td>
<td>50 000 000</td>
<td>2.348</td>
</tr>
</tbody>
</table>
from fc_hypermesh import bench
bench(3, range(40, 190, 20), type='simplex', box=[[-1, 1], [-1, 1], [-1, 1]], order=1)

Listing 24: Tessellation of $[-1, 1]^3$ by simplices

<table>
<thead>
<tr>
<th>$N$</th>
<th>$n_q$</th>
<th>$n_{me}$</th>
<th>Python</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>68 921</td>
<td>384 000</td>
<td>0.188</td>
</tr>
<tr>
<td>60</td>
<td>226 981</td>
<td>1 296 000</td>
<td>0.26</td>
</tr>
<tr>
<td>80</td>
<td>531 441</td>
<td>3 072 000</td>
<td>0.348</td>
</tr>
<tr>
<td>100</td>
<td>1 030 301</td>
<td>6 000 000</td>
<td>0.473</td>
</tr>
<tr>
<td>120</td>
<td>1 771 561</td>
<td>10 368 000</td>
<td>0.684</td>
</tr>
<tr>
<td>140</td>
<td>2 803 221</td>
<td>16 464 000</td>
<td>0.957</td>
</tr>
<tr>
<td>160</td>
<td>4 173 281</td>
<td>24 576 000</td>
<td>1.324</td>
</tr>
<tr>
<td>180</td>
<td>5 929 741</td>
<td>34 992 000</td>
<td>1.918</td>
</tr>
</tbody>
</table>

Table 12: Tessellation of $[-1, 1]^3$ by simplices

from fc_hypermesh import bench
bench(4, [10, 20, 25, 30, 35, 40], type='simplex', order=1, box=[[-1, 1], [-1, 1], [-1, 1], [-1, 1]])

Listing 25: Tessellation of $[-1, 1]^4$ by simplices

<table>
<thead>
<tr>
<th>$N$</th>
<th>$n_q$</th>
<th>$n_{me}$</th>
<th>Python</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>14 641</td>
<td>240 000</td>
<td>0.276</td>
</tr>
<tr>
<td>20</td>
<td>194 481</td>
<td>3 840 000</td>
<td>0.555</td>
</tr>
<tr>
<td>25</td>
<td>456 976</td>
<td>9 375 000</td>
<td>0.98</td>
</tr>
<tr>
<td>30</td>
<td>923 521</td>
<td>19 440 000</td>
<td>1.768</td>
</tr>
<tr>
<td>35</td>
<td>1 679 616</td>
<td>36 015 000</td>
<td>3.039</td>
</tr>
</tbody>
</table>

Table 13: Tessellation of $[-1, 1]^4$ by simplices

from fc_hypermesh import bench
bench(5, range(2, 14, 2), type='simplex', order=1, box=[[-1, 1], [-1, 1], [-1, 1], [-1, 1], [-1, 1]])

Listing 26: Tessellation of $[-1, 1]^5$ by simplices

<table>
<thead>
<tr>
<th>$N$</th>
<th>$n_q$</th>
<th>$n_{me}$</th>
<th>Python</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>243</td>
<td>3 840</td>
<td>0.42</td>
</tr>
<tr>
<td>4</td>
<td>3 125</td>
<td>122 880</td>
<td>0.41</td>
</tr>
<tr>
<td>6</td>
<td>16 807</td>
<td>933 120</td>
<td>0.556</td>
</tr>
<tr>
<td>8</td>
<td>59 049</td>
<td>3 932 160</td>
<td>0.794</td>
</tr>
<tr>
<td>10</td>
<td>161 051</td>
<td>12 000 000</td>
<td>1.449</td>
</tr>
<tr>
<td>12</td>
<td>371 293</td>
<td>29 859 840</td>
<td>3.079</td>
</tr>
</tbody>
</table>

Table 14: Tessellation of $[-1, 1]^5$ by simplices

6 References


Informations for developers/maintainers of the Python package

<table>
<thead>
<tr>
<th>Package</th>
<th>Tag</th>
<th>Commit</th>
<th>Date</th>
<th>Time</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>fc-hypermesh</td>
<td>0.1.1</td>
<td>97ab31d0ba3bbaef0095bd487bcb11190a309529f0</td>
<td>2019-12-29</td>
<td>14:51-49</td>
<td>0</td>
</tr>
<tr>
<td>fc-bench</td>
<td>0.2.0</td>
<td>56ba49013b1836cc812c9ae4c6f15699966fdd5</td>
<td>2019-12-23</td>
<td>07:53-49</td>
<td>0</td>
</tr>
<tr>
<td>fc-tools</td>
<td>0.0.24</td>
<td>2ae83c0d68196297117b005ad88ab3288725c</td>
<td>2019-12-21</td>
<td>11:34-49</td>
<td>0</td>
</tr>
<tr>
<td>fctools</td>
<td></td>
<td>c7bea2d6b67a5391d17097463e6f3541282</td>
<td>2019-03-22</td>
<td>12:57-26</td>
<td>True</td>
</tr>
<tr>
<td>fc-config</td>
<td></td>
<td>533e20f025b614ee3d91a31521d38edf6e629c</td>
<td>2019-12-30</td>
<td>07:31-28</td>
<td>True</td>
</tr>
</tbody>
</table>