FC_HYPERMESH package, User's Guide *

François Cuvelier†

March 4, 2017

Abstract

This object-oriented Python package allows to mesh any d-orthotopes (hyperrectangle in dimension d) and their m-faces by simplices or orthotopes. It was created to show the implementation of the algorithms of [1]. The FC_HYPERMESH package uses Python objects and is provided with meshes visualisation tools for dimension leather or equal to 3.

Contents

---

*Compiled with Python 3.6.0, packages FC_HYPERMESH-0.0.3 and `fc_tools`-0.0.5

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

This work was partially supported by ANR Dedales.

1 Introduction

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

In the folowing 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 Contents of the fc_hypermesh package

2.1 Class object `EltMesh`

An elementary mesh class object `EltMesh` is used to store only one mesh, the main mesh as well as any of the meshes of the m-faces. 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<=m<=d, m == d for the main mesh.
- **type** : 0 for simplicial mesh or 1 for orthotope mesh.
- 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)

2.2 Class object `OrthMesh`

The aim of the class object `OrthMesh` is to efficiently create an object which contains a mesh of a d-orthotope and all its m-face meshes.

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

- d: space dimension
- `type`: string 'simplicial' or 'orthotope' mesh
- 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 $(\mathrm{d}-1)$-faces, Faces[1] is a list of all the meshes of the $(\mathrm{d}-2)$-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.

2.2.1 Constructor

The `OrthMesh` constructor is :

$$\text{Oh} = \text{OrthMesh(d,N)}$$

where N is either a 1-by-d array/list such that N[i−1] is the number of discretization for $[a_i, b_i]$ or either an integer if the the number of discretization is the same in all space directions.
Some options are proposed with the constructor:

$$\text{Oh} = \text{OrthMesh(d,N,key=value)}$$

- box = value : where value is a d-by-2 list or array such that value[i−1][0] is a_i value and value[i−1][1] is b_i value. Default is $[0, 1]^{\mathrm{d}}$.
- `type = value` : The default value for optional key parameter `type` is 'simplicial' and ortherwise 'orthotope' can be used.

2.2.2 plotmesh property

$$\text{plotmesh()}$$

Used matplotlib to represents the mesh represented by an `OrthMesh` object. Some options are proposed with this property:

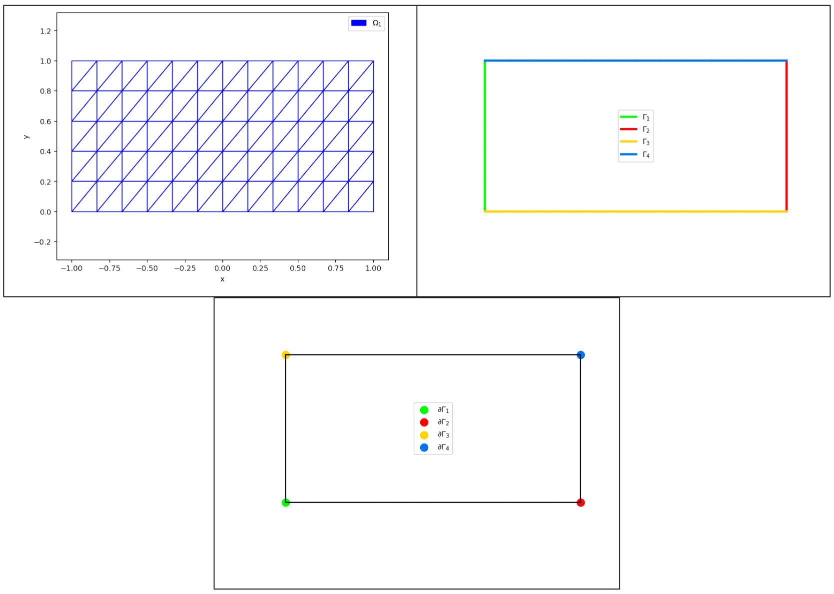
$$\text{plotmesh(key=value)}$$

- legend = value : if value is True, a legend is displayed. Default is False.
- m = value : plots all the m-faces of the mesh. Default m = d i.e. the main mesh. (0 ⩽m⩽d)
- ...

3 Using the fc_hypermesh package

3.1 2d-orthotope meshing by simplices

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

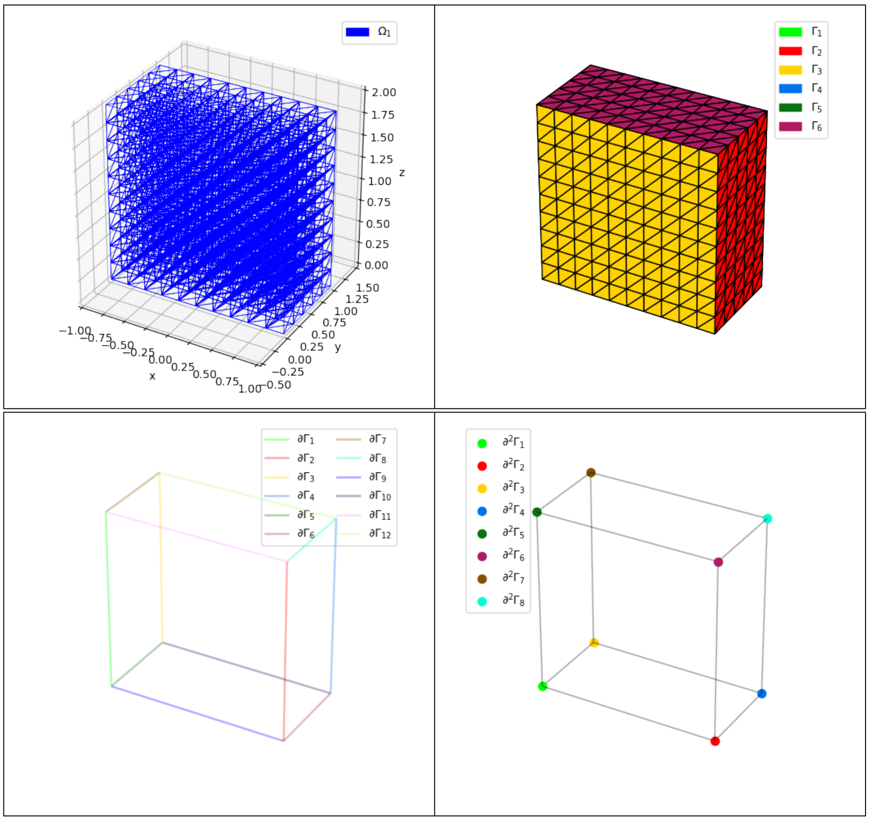



```python
oTh=OrthMesh(2,[12,5],type='simplicial',box=[[-1,1],[0,1]])
plt.figure(1)
oTh.plotmesh(legend=True)
set_axes_equal()
plt.figure(2)
oTh.plotmesh(m=1,legend=True,linewidth=3)
plt.axis('off')
set_axes_equal()
plt.figure(3)
oTh.plotmesh(m=1,color='black')
oTh.plotmesh(m=0,legend=True,s=105)
plt.axis('off')
set_axes_equal()
DisplayFigures()
```

Listing 1: 2D simplicial **`OrthMesh`** object with Python 3.6.0, main mesh (upper left), 1-face meshes (upper right), and 0-face meshes (bottom)

3.2 3d-orthotope meshing by simplices

In Listing 2, an **`OrthMesh`** object is built under Python 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.

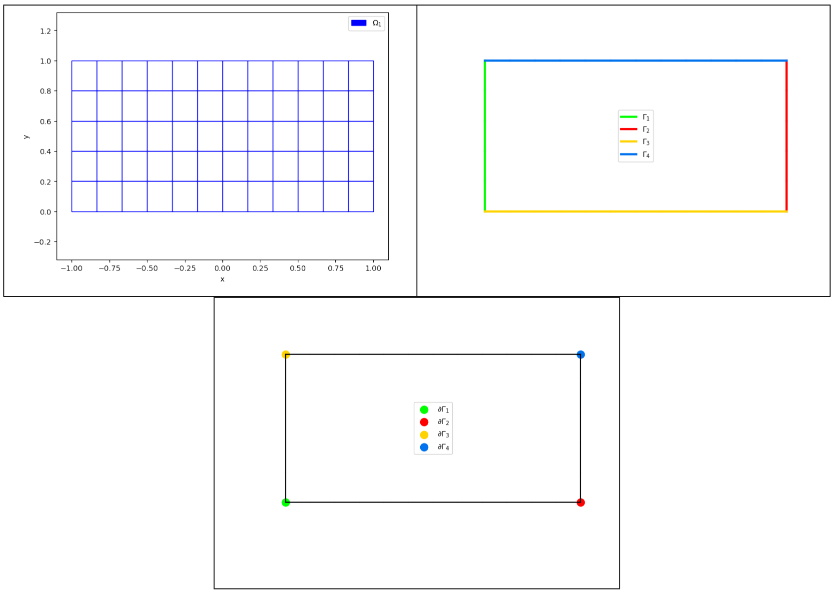



```
oTh=OrthMesh(3,[10,5,10],box=[[-1,1],[0,1],[0,2]])
plt.figure(1)
oTh.plotmesh(legend=True,linewidth=0.5)
set_axes_equal()
plt.figure(2)
oTh.plotmesh(m=2,legend=True,edgecolor=[0,0,0])
plt.axis('off')
set_axes_equal()
plt.figure(3)
oTh.plotmesh(m=2,edgecolor=[0,0,0],color='none')
oTh.plotmesh(m=1,legend=True,linewidth=2,alpha=0.3)
plt.axis('off')
set_axes_equal()
plt.figure(4)
oTh.plotmesh(m=1,color='black',alpha=0.3)
oTh.plotmesh(m=0,legend=True,s=55)
set_axes_equal()
plt.axis('off')
```

Listing 2: 3D simplicial `OrthMesh` object with Python 3.6.0, main mesh (upper left), 2-face meshes (upper right), 1-face meshes (bottom left) and 0-face meshes (bottom right)

3.3 2d-orthotope meshing by orthotopes

In Listing 3, an `OrthMesh` object is built under Pythonfor the orthotope $[-1,1]\times[0,1]$ with orthotope elements and $\mathbf{N}=(10,5,10)$. The main mesh and all the m-face meshes of the resulting object are plotted.

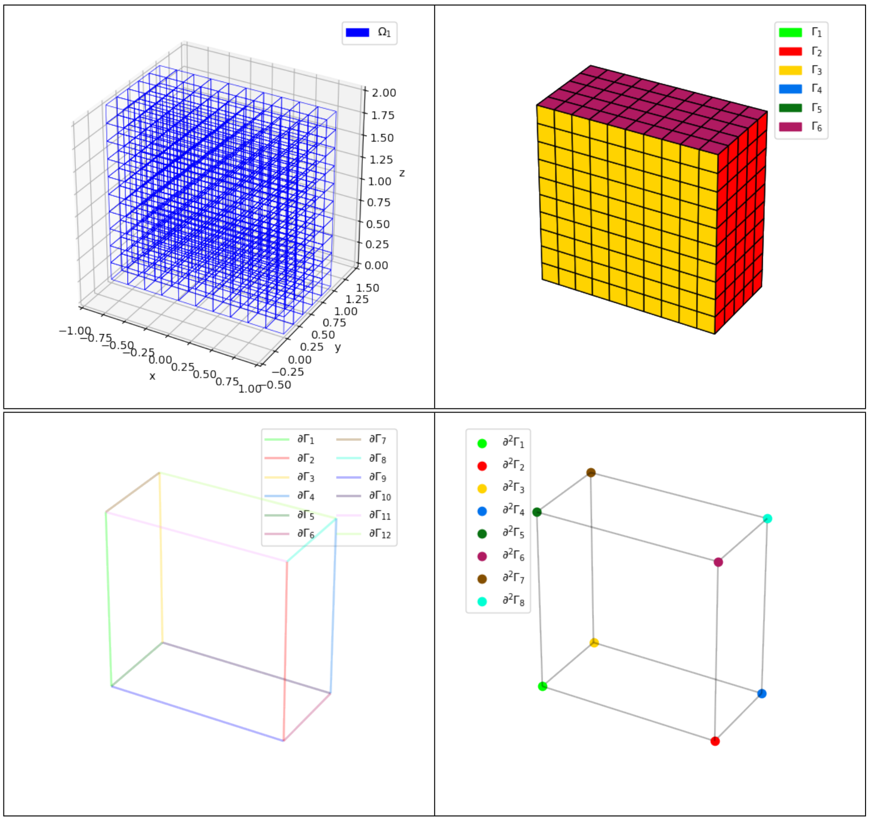



```
oTh=OrthMesh(2,[12,5],type='orthotope',box=[[-1,1],[0,1]])
plt.figure(1)
oTh.plotmesh(legend=True)
set_axes_equal()
plt.figure(2)
oTh.plotmesh(m=1,legend=True,linewidth=3)
plt.axis('off')
set_axes_equal()
plt.figure(3)
oTh.plotmesh(m=1,color='black')
oTh.plotmesh(m=0,legend=True,s=105)
plt.axis('off')
set_axes_equal()
```

Listing 3: 2D orthotope `OrthMesh` object with Python 3.6.0, main mesh (upper left), 1-face meshes (upper right), and 0-face meshes (bottom)

3.4 3d-orthotope meshing by orthotopes

In Listing 4, an `OrthMesh` object is built under Pythonfor 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.




```
oTh=OrthMesh(3,[10,5,10],type='orthotope',
      box=[[-1,1],[0,1],[0,2]])
plt.figure(1)
oTh.plotmesh(legend=True,linewidth=0.5)
set_axes_equal()
plt.figure(2)
oTh.plotmesh(m=2,legend=True,edgecolor=[0,0,0])
plt.axis('off')
set_axes_equal()
plt.figure(3)
oTh.plotmesh(m=2,edgecolor=[0,0,0],color='none')
oTh.plotmesh(m=1,legend=True,linewidth=2,alpha=0.3)
plt.axis('off')
set_axes_equal()
plt.figure(4)
oTh.plotmesh(m=1,color='black',alpha=0.3)
oTh.plotmesh(m=0,legend=True,s=55)
set_axes_equal()
plt.axis('off')
```

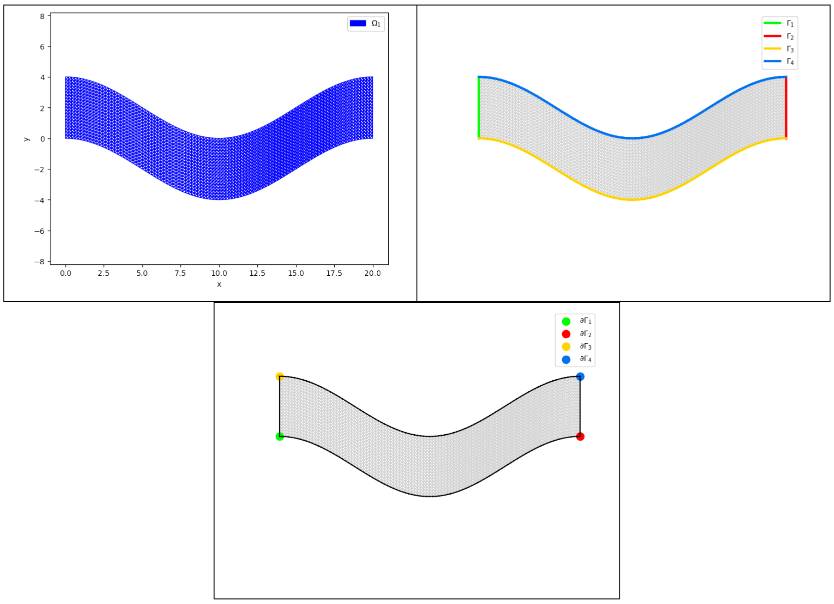
Listing 4: 3D orthotope `OrthMesh` object with Python 3.6.0, main mesh (upper left), 2-face meshes (upper right), 1-face meshes (bottom left) and 0-face meshes (bottom right)

3.5 Mapping of a 2d-orthotope meshing by simplices

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

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

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

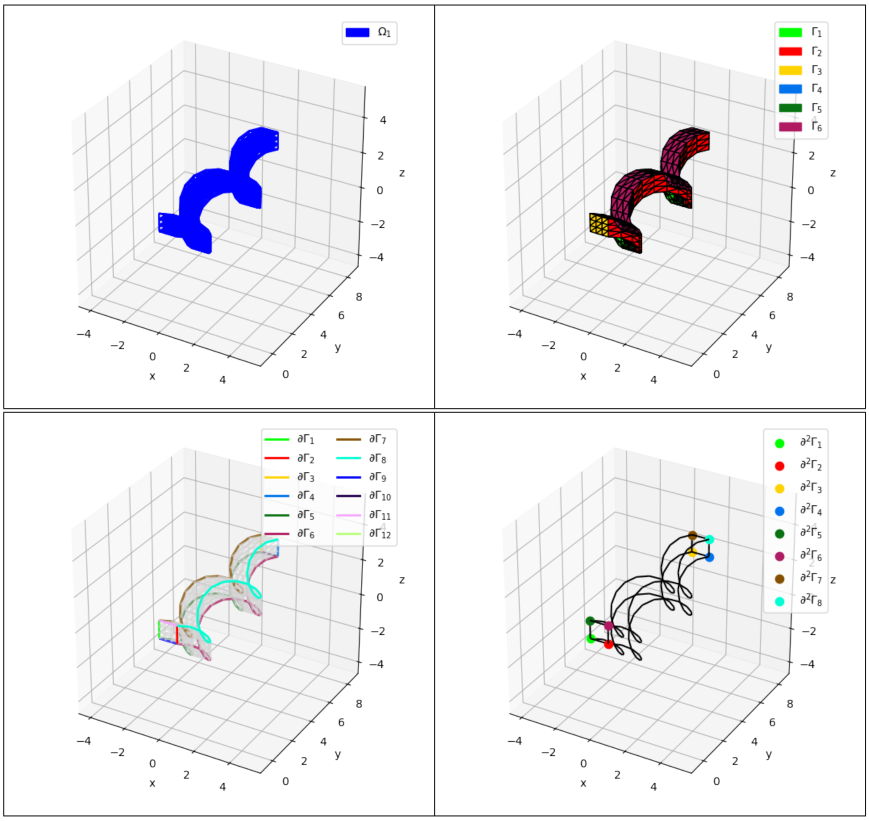


```python
import numpy as np
trans=lambda q:
    np.array([20*q[0],2*(2*q[1]-1+np.cos(2*np.pi*q[0]))])
oTh=OrthMesh(2,[100,20],type='simplicial',mapping=trans)
plt.figure(1)
oTh.plotmesh(legend=True)
plt.axis('equal')
plt.figure(2)
oTh.plotmesh(color='lightgray')
oTh.plotmesh(m=1,legend=True,linewidth=3)
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 5: Mapping of a 2D simplicial `OrthMesh` object with Python 3.6.0, main mesh (upper left), 1-face meshes (upper right), and 0-face meshes (bottom)

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

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



```
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='simplicial',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)
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 6: Mapping of a 3D orthotope `OrthMesh` object with Python 3.6.0, main mesh (upper left), 2-face meshes (upper right), 1-face meshes (bottom left) and 0-face meshes (bottom right)

4 Memory consuming

Take care when using theses codes with memory consuming : the number of points $\mathrm{n_q}$ 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

$$\mathrm{n_q} = (N+1)^d, \quad \text{for both tessellation and triangulation}$$

and

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

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

$$\text{mem. size of q} = \mathrm{d} \times \mathrm{n_q} \times 8 \text{ octets}$$

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

$$\text{mem. size of me} = \begin{cases} 2^{\mathrm{d}} \times \mathrm{n_{me}} \times 4 \text{ octets} & \text{(tessellation by orthotopes)} \\ (\mathrm{d}+1) \times \mathrm{n_{me}} \times 4 \text{ octets} & \text{(tessellation by simplices)} \end{cases}$$

For $N = 10$ and $\mathrm{d} \in [\![1, 8]\!]$, the values of $\mathrm{n_q}$ and $\mathrm{n_{me}}$ are given in Table 1. The memory usage for the corresponding array q and array me is available in Table 2.

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

Table 1: Number of vertices $\mathrm{n_q}$ and number of elements $\mathrm{n_{me}}$ 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	88 o	80 o	80 o
2	1 ko	1 ko	2 ko
3	31 ko	32 ko	96 ko
4	468 ko	640 ko	4 Mo
5	6 Mo	12 Mo	288 Mo
6	85 Mo	256 Mo	20 Go
7	1 Go	5 Go	1 612 Go
8	13 Go	102 Go	145 152 Go

Table 2: 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 $N = 10$.

5 Benchmarks

For all the following tables, the computational costs of the `OrthMesh` constructor are given for the orthotope $[-1,1]^d$ under Python 3.6.0. The computations were done on a laptop with `Core i7-4800MQ` processor and 16Go of RAM under Ubuntu 14.04 LTS (64bits).

In the following pages, computational costs of the `OrthMesh` constructor will be done by using bench_gen function. As sample, we give an example with output. Thereafter, all the output will be presented in tabular form.

```
from fc_hypermesh.benchs import bench_gen
bench_gen(3,'simplicial',[[-1,1],[-1,1],[-1,1]],
    range(20,170,20))
```

Listing 7: bench sample

Output

```
# BENCH in dimension 3 with simplicial mesh
#d: 3
#type: simplicial
#box: [[-1, 1], [-1, 1], [-1, 1]]
#desc:  N        nq        nme    time(s)
       20      9261      48000     0.169
       40     68921     384000     0.182
       60    226981    1296000     0.217
       80    531441    3072000     0.339
      100   1030301    6000000     0.438
      120   1771561   10368000     0.629
      140   2803221   16464000     0.896
      160   4173281   24576000     1.272
```

5.1 Tessellation by orthotopes

```
from fc_hypermesh.benchs import  bench_gen
bench_gen(2,'orthotope',[[-1,1],[-1,1]],range(1000,6000,1000))
```

Listing 8: Tessellation of $[-1,1]^2$ by orthotopes

N	n_q	n_{me}	Python
1000	1 002 001	1 000 000	0.138
2000	4 004 001	4 000 000	0.248
3000	9 006 001	9 000 000	0.441
4000	16 008 001	16 000 000	0.722
5000	25 010 001	25 000 000	1.081

Table 3: Tessellation of $[-1,1]^2$ by orthotopes

```python
from fc_hypermesh.benchs import bench_gen
bench_gen(3,'orthotope',[[-1,1],[-1,1],[-1,1]],
    range(50,400,50))
```

Listing 9: Tessellation of $[-1,1]^3$ by orthotopes

N	n_q	n_{me}	Python
50	132 651	125 000	0.168
100	1 030 301	1 000 000	0.229
150	3 442 951	3 375 000	0.368
200	8 120 601	8 000 000	0.684
250	15 813 251	15 625 000	1.143
300	27 270 901	27 000 000	2.009
350	43 243 551	42 875 000	3.142

Table 4: Tessellation of $[-1,1]^3$ by orthotopes

```python
from fc_hypermesh.benchs import bench_gen
bench_gen(4,'orthotope',[[-1,1],[-1,1],[-1,1],[-1,1]],
    [10,20,30,40,50,62])
```

Listing 10: Tessellation of $[-1,1]^4$ by orthotopes

N	n_q	n_{me}	Python
10	14 641	10 000	0.26
20	194 481	160 000	0.258
30	923 521	810 000	0.335
40	2 825 761	2 560 000	0.521
50	6 765 201	6 250 000	0.909
62	15 752 961	14 776 336	1.818

Table 5: Tessellation of $[-1,1]^4$ by orthotopes

```python
from fc_hypermesh.benchs import bench_gen
bench_gen(5,'orthotope',[[-1,1],[-1,1],[-1,1],[-1,1],[-1,1]],
    [5,10,15,20,25,27])
```

Listing 11: Tessellation of $[-1,1]^5$ by orthotopes

N	n_q	n_{me}	Python
5	7 776	3 125	0.5
10	161 051	100 000	0.503
15	1 048 576	759 375	0.653
20	4 084 101	3 200 000	1.122
25	11 881 376	9 765 625	2.507
27	17 210 368	14 348 907	3.353

Table 6: Tessellation of $[-1,1]^5$ by orthotopes

5.2 Tessellation by simplices

```
from fc_hypermesh.benchs import bench_gen
bench_gen(2,'simplicial',[[-1,1],[-1,1]],range(1000,6000,1000))
```

Listing 12: Tessellation of $[-1,1]^2$ by simplices

N	n_q	n_{me}	Python
1000	1 002 001	2 000 000	0.168
2000	4 004 001	8 000 000	0.371
3000	9 006 001	18 000 000	0.731
4000	16 008 001	32 000 000	1.214
5000	25 010 001	50 000 000	1.817

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

```
from fc_hypermesh.benchs import bench_gen
bench_gen(3,'simplicial',[[-1,1],[-1,1],[-1,1]],
    range(40,190,20))
```

Listing 13: Tessellation of $[-1,1]^3$ by simplices

N	n_q	n_{me}	Python
40	68 921	384 000	0.192
60	226 981	1 296 000	0.235
80	531 441	3 072 000	0.329
100	1 030 301	6 000 000	0.454
120	1 771 561	10 368 000	0.656
140	2 803 221	16 464 000	0.952
160	4 173 281	24 576 000	1.332
180	5 929 741	34 992 000	1.833

Table 8: Tessellation of $[-1,1]^3$ by simplices

```
from fc_hypermesh.benchs import bench_gen
bench_gen(4,'simplicial',[[-1,1],[-1,1],[-1,1],[-1,1]],
    [10,20,25,30,35,40])
```

Listing 14: Tessellation of $[-1,1]^4$ by simplices

N	n_q	n_{me}	Python
10	14 641	240 000	0.311
20	194 481	3 840 000	0.566
25	456 976	9 375 000	0.968
30	923 521	19 440 000	1.809
35	1 679 616	36 015 000	3.015

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

```python
from fc_hypermesh.benchs import bench_gen
bench_gen(5,'simplicial',[[-1,1],[-1,1],[-1,1],[-1,1],[-1,1]],
    range(2,14,2))
```

Listing 15: Tessellation of $[-1, 1]^5$ by simplices

N	n_q	n_{me}	Python
2	243	3 840	0.543
4	3 125	122 880	0.544
6	16 807	933 120	0.58
8	59 049	3 932 160	0.694
10	161 051	12 000 000	1.347
12	371 293	29 859 840	2.885

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

References

[1] F. Cuvelier and G. Scarella. Vectorized algorithms for regular tessellations of d-orthotopes and their faces. http://www.math.univ-paris13.fr/ cuvelier/-docs/reports/HyperMesh/HyperMesh_0.0.4.pdf, 2016.