



FC_HYPERMESH package, User's Guide *

François Cuvelier[†] Gilles Scarella[‡]

March 7, 2018

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

1	Introduction	2
2	Installation and uninstall	2
3	Contents of the FC_HYPERMESH package	3
3.1	Class object <code>EltMesh</code>	3
3.2	Class object <code>OrthMesh</code>	4
3.2.1	Constructor	4
3.2.2	<code>plotmesh</code> method	5
4	Using the FC_HYPERMESH package	5
4.1	2d-orthotope meshing by simplices	5
4.2	3d-orthotope meshing by simplices	6
4.3	2d-orthotope meshing by orthotopes	8
4.4	3d-orthotope meshing by orthotopes	8

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

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

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

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

4.5	Mapping of a 2d-orthotope meshing by simplices	10
4.6	Mapping of a 3d-orthotope meshing by orthotopes	11
5	Memory consuming	13
6	Benchmarks	14
6.1	Tessellation by orthotopes	14
6.2	Tessellation by simplices	16

1 Introduction

The `FC_HYPERMESH` package contains a simple class object `OrthMesh` which permits, in any dimension $d \geq 1$, to obtain a simplicial mesh or orthotope mesh with all their m -faces, $0 \leq m < d$, of a d -*orthotope*. If the Matplotlib package is installed, it is also possible with the `plotmesh` method of the class object `OrthMesh` to represent a mesh or its m -faces for $d \leq 3$.

This package is referenced on PyPI, the Python Package Index, and was tested under Python and Miniconda distributions:

- **Windows 10**
 - with Miniconda Python 2.7 distribution
 - with Miniconda Python 3.6 distribution
- **Ubuntu 14.04.5 LTS**
 - with Python 2.7.13 compiled from source
 - with Python 3.6.0 compiled from source
 - with Miniconda Python 2.7 distribution
 - with Miniconda Python 3.6 distribution
- **MacOS Sierra**
 - with Miniconda Python 2.7 distribution
 - with Miniconda Python 3.6 distribution

In a first step, the installation of this package is presented .. Thereafter the class object `OrthMesh` is described and some examples are provided. Then 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 $d \in \llbracket 1, 5 \rrbracket$.

2 Installation and uninstall

Installation :

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

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

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

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

An other 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:

```
$ 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:

```
$ 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 Contents of the `fc_hypermesh` package

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

3.1 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, $0 \leq m \leq d$, $m == d$ 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)

3.2 Class object `OrthMesh`

The aim of the class object `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.

Let the d-orthotope defined by $[a_1, b_1] \times \dots \times [a_d, b_d]$. The class object `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 '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 $(d - 1)$ -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.

3.2.1 Constructor

The `OrthMesh` constructor is :

```
Oh = 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:

```
Oh = 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 $[0, 1]^d$.
- **type = value** : The default **value** for optional key parameter **type** is 'simplicial' and otherwise 'orthotope' can be used.

3.2.2 plotmesh method

The `plotmesh()` member function uses Matplotlib Python package [2, 3] to represent the mesh given by an `OrthMesh` object. Some options are proposed with this function:

`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. ($0 \leq m \leq d$)
- ...

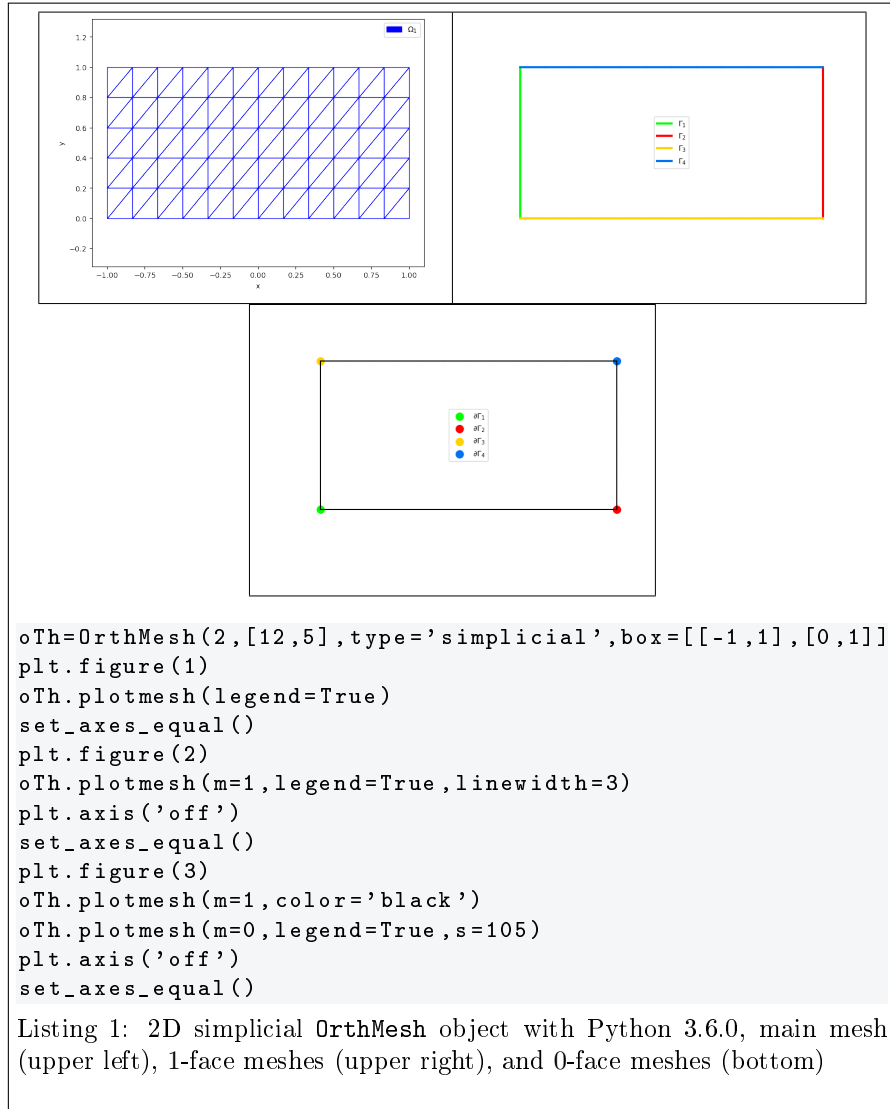
4 Using the `fc_hypermesh` package

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

```
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
```

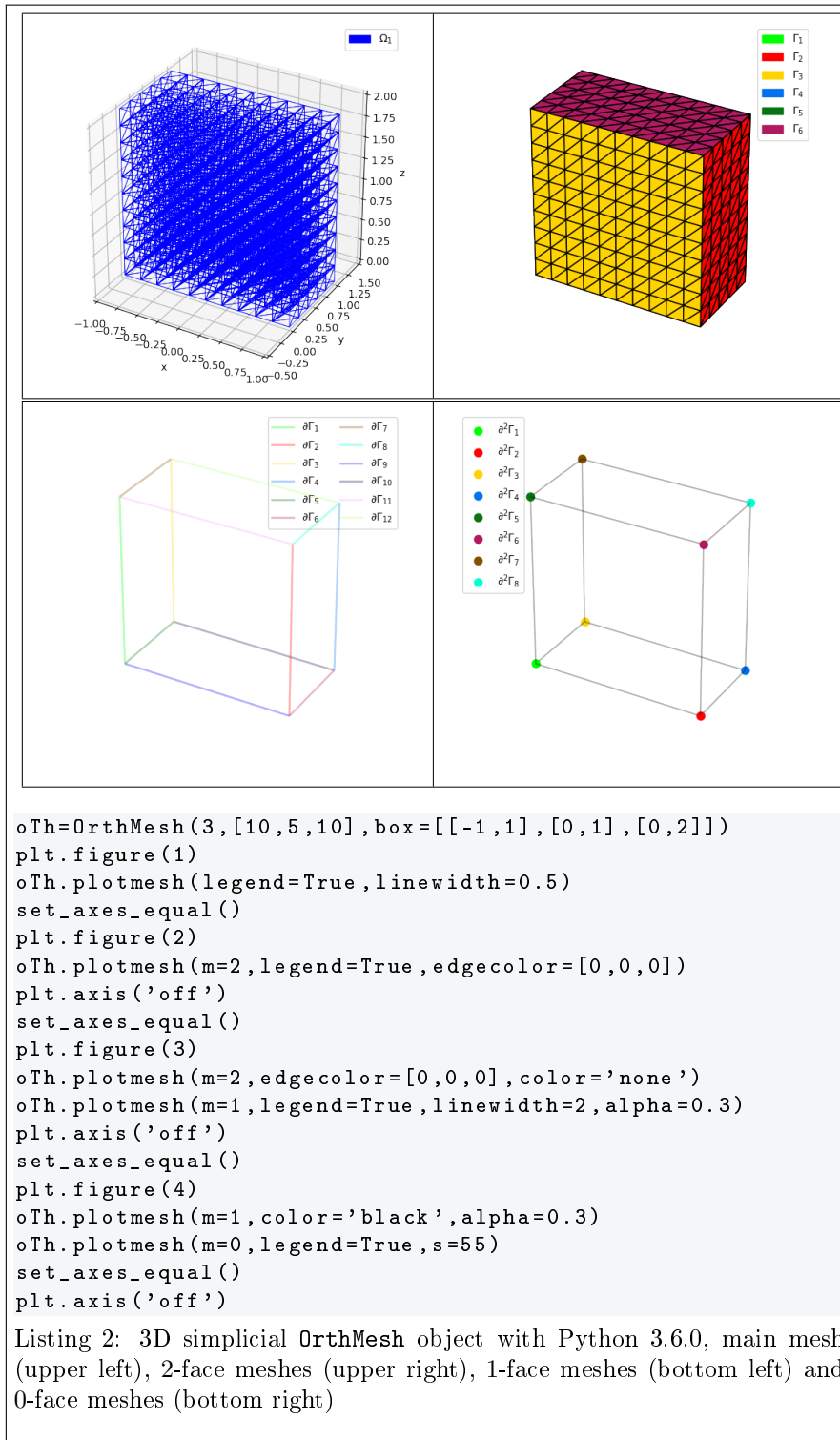
4.1 2d-orthotope meshing by simplices

In Listing 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.



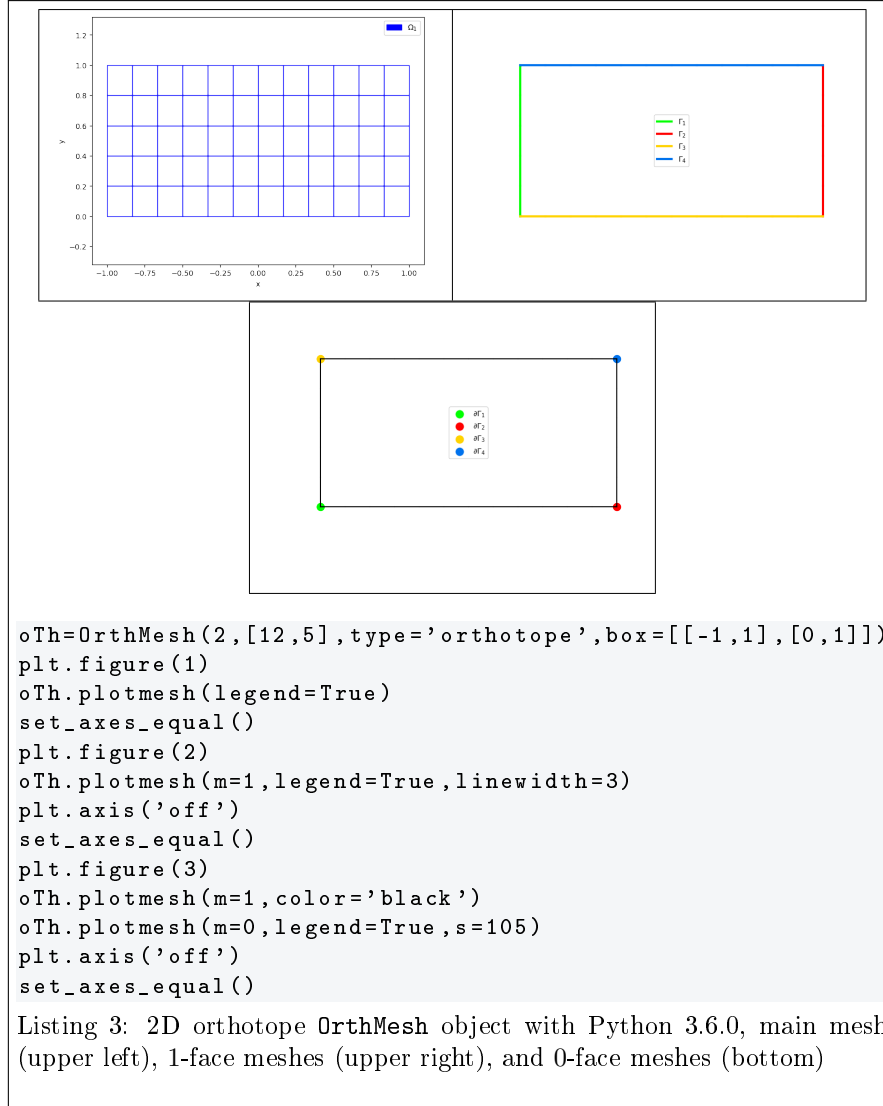
4.2 3d-orthotope meshing by simplices

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.



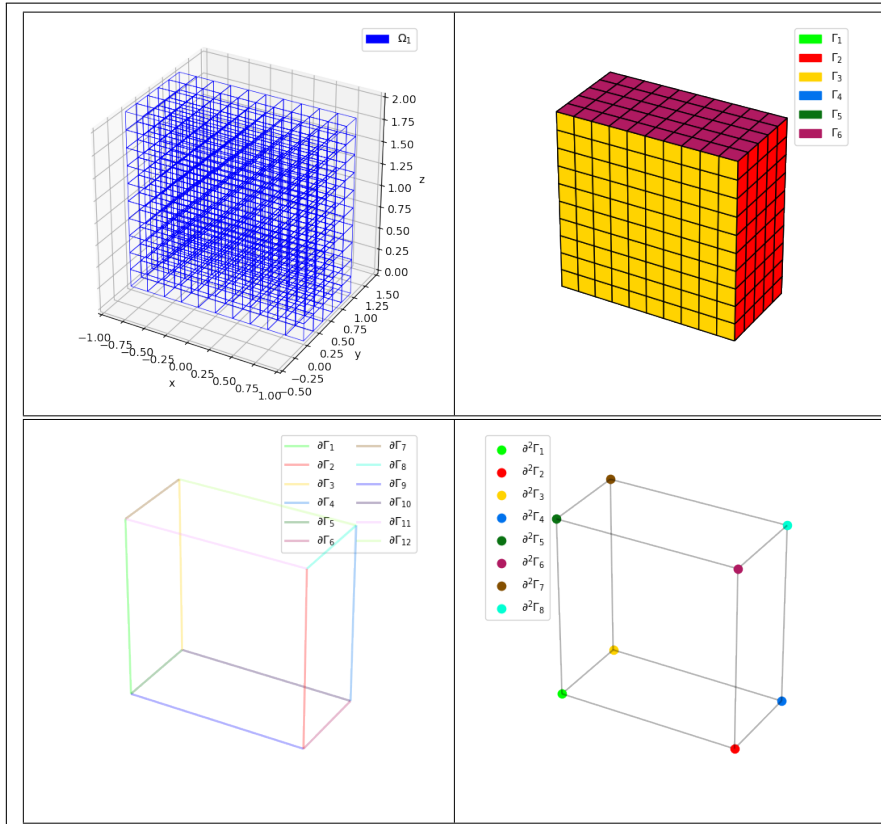
4.3 2d-orthotope meshing by orthotopes

In Listing 3, an `OrthMesh` object is built under Python for 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.



4.4 3d-orthotope meshing by orthotopes

In Listing 4, 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.



```

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')

```

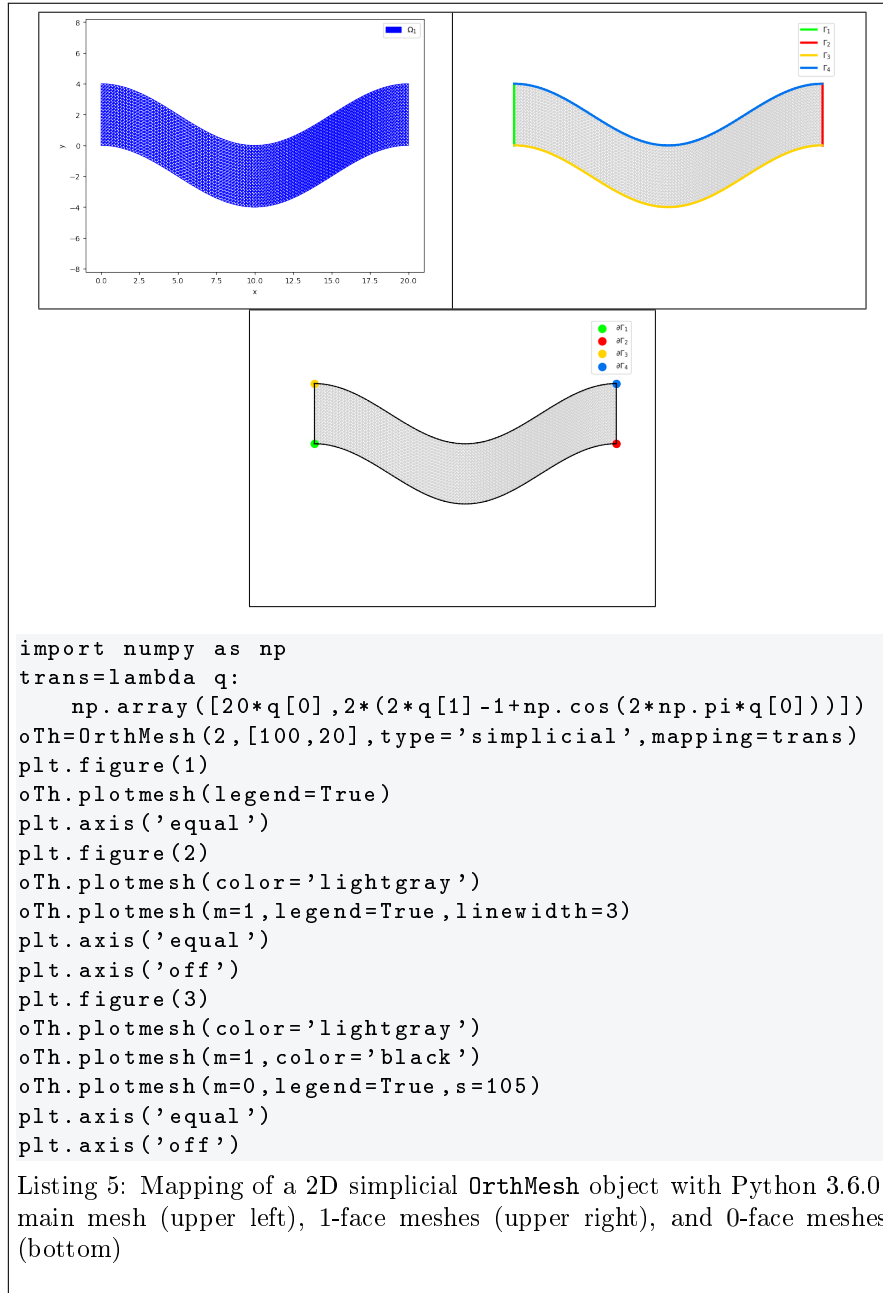
Listing 4: 3D orthotope `OrthMesh` object with Python 3.6.0, main mesh (upper left), 2-face meshes (upper right), 1-face meshes (bottom left) and 0-face meshes (bottom right)

4.5 Mapping of a 2d-orthotope meshing by simplices

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

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

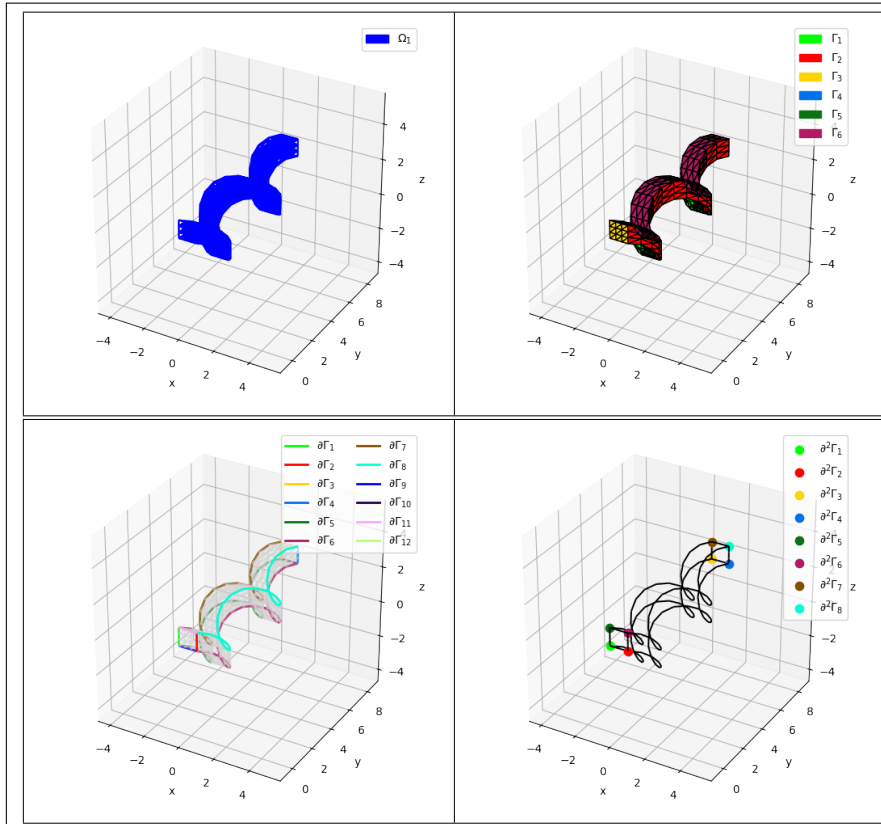
$$\begin{pmatrix} x \\ y \end{pmatrix} \longrightarrow F(x, y) = \begin{pmatrix} 20x \\ 2(2y - 1 + \cos(2\pi x)) \end{pmatrix}$$



4.6 Mapping of a 3d-orthotope meshing by orthotopes

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

$$\begin{aligned} [0, 1] \times [0, 1] \times [0, 1] &\longrightarrow \mathbb{R}^3 \\ \begin{pmatrix} x \\ y \\ z \end{pmatrix} &\longrightarrow F(x, y, z) = \begin{pmatrix} x + \sin(4\pi y) \\ 10y \\ z + \cos(4\pi y) \end{pmatrix} \end{aligned}$$



```

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)

5 Memory consuming

Take care when using theses 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

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

and

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

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

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

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

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

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

d	$n_q = (N + 1)^d$	$n_{me} = N^d$ (orthotopes)	$n_{me} = d!N^d$ (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 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 $N = 10$.

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 $N = 10$.

6 Benchmarks

For all the following tables, the computational costs of the `OrthMesh` constructor are given for the orthotope $[-1, 1]^d$ under Python 3.6.0. 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 `bench01` function. As sample, we give an example with output. Thereafter, all the output will be presented in tabular form.

```
from fc_hypermesh import bench01
bench01(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      0.202
      40     68921     384000      0.204
      60    226981    1296000      0.266
      80    531441    3072000      0.353
     100   1030301    6000000      0.478
     120   1771561   10368000      0.671
     140   2803221   16464000      0.941
     160   4173281   24576000      1.315
```

6.1 Tessellation by orthotopes

```
from fc_hypermesh import bench01
bench01(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	0.167
2000	4 004 001	4 000 000	0.266
3000	9 006 001	9 000 000	0.457
4000	16 008 001	16 000 000	0.738
5000	25 010 001	25 000 000	1.127

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

```

from fc_hypermesh import bench01
bench01(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	0.197
100	1 030	301		1 000 000	0.257
150	3 442	951		3 375 000	0.413
200	8 120	601		8 000 000	0.696
250	15 813	251		15 625 000	1.172
300	27 270	901		27 000 000	2.038
350	43 243	551		42 875 000	3.14

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

```

from fc_hypermesh import bench01
bench01(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	0.288
20	194	481		160 000	0.297
30	923	521		810 000	0.38
40	2 825	761		2 560 000	0.565
50	6 765	201		6 250 000	0.994
62	15 752	961		14 776 336	1.854

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

```

from fc_hypermesh import bench01
bench01(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	0.461
10	161	051		100 000	0.476
15	1 048	576		759 375	0.681
20	4 084	101		3 200 000	1.125
25	11 881	376		9 765 625	2.344
27	17 210	368		14 348 907	3.166

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

6.2 Tessellation by simplices

```
from fc_hypermesh import bench01
bench01(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	0.196
2000	4 004 001	8 000 000	0.393
3000	9 006 001	18 000 000	0.74
4000	16 008 001	32 000 000	1.222
5000	25 010 001	50 000 000	1.885

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

```
from fc_hypermesh import bench01
bench01(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	0.21
60	226 981	1 296 000	0.269
80	531 441	3 072 000	0.359
100	1 030 301	6 000 000	0.487
120	1 771 561	10 368 000	0.684
140	2 803 221	16 464 000	0.959
160	4 173 281	24 576 000	1.337
180	5 929 741	34 992 000	1.804

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

```
from fc_hypermesh import bench01
bench01(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	0.304
20	194 481	3 840 000	0.555
25	456 976	9 375 000	0.974
30	923 521	19 440 000	1.697
35	1 679 616	36 015 000	2.887

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


```

from fc_hypermesh import bench01
bench01(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		0.466
4	3	125		122	880		0.452
6	16	807		933	120		0.561
8	59	049		3	932	160	0.696
10	161	051		12	000	000	1.314
12	371	293		29	859	840	2.863

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.
- [2] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.
- [3] Matplotlib team. Matplotlib, 2017. User’s Guide.