



FC-HYPERMESH Octave package, User's Guide *

François Cuvelier[†]

October 30, 2017

Abstract

This object-oriented Octave 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 vectorized algorithms presented in []. The FC-HYPERMESH package uses Octave objects and is provided with meshes visualisation tools for dimension leather or equal to 3.

Contents

1	Introduction	2
2	Installation	2
2.1	Installation via pkg command (recommanded)	2
2.2	Installation automatic, all in one	3
3	Using the FC-HYPERMESH package	3
3.1	Class object <code>OrthMesh</code>	4
3.1.1	Constructor	4
3.1.2	<code>plotmesh</code> method	5
3.2	2d-orthotope meshing by simplices	5
3.3	3d-orthotope meshing by simplices	6
3.4	2d-orthotope meshing by orthotopes	8
3.5	3d-orthotope meshing by orthotopes	8

*Compiled with Octave 4.2.1

[†]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 supported by the ANR project DEDALES under grant ANR-14-CE23-0005.

3.6	Mapping of a 2d-orthotope meshing by simplices	10
3.7	3d-orthotope meshing by orthotopes	11

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, $0 \leq m < d$. It is also possible with the method function `plot` of the class object `OrthMesh` to represent a mesh or its m -faces for $d \leq 3$.

This package was tested under

Windows 10: with Octave 4.2.1

MacOS Sierra: with Octave 4.2.1 (install with homebrew)

Ubuntu 14.04.5 LTS: with Octave 4.2.1 and 4.2.0 (compiled from source)

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

In the following section, the class object `OrthMesh` is presented. Thereafter some warning statements on the memory used by these objects in high dimension are given. Finally computation times for orthotope meshes and simplicial meshes are given in dimension $d \in \llbracket 1, 5 \rrbracket$.

2 Installation

2.1 Installation via pkg command (recommended)

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

```
# wget http://www.math.univ-paris13.fr/~cuvelier/software/codes/
  Octave/fc-tools/0.0.18/fc-tools-0.0.18.tar.gz
# wget http://www.math.univ-paris13.fr/~cuvelier/software/codes/
  Octave/fc-hypermesh/0.0.4/fc-hypermesh-0.0.4.tar.gz
```

- Under Octave :

```
>> pkg install fc-tools-0.0.18.tar.gz
>> pkg install fc-hypermesh-0.0.4.tar.gz
```

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

```
>> pkg load fc-hypermesh
```

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

```
>> fc_hypermesh_demo01
```

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

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

2.2 Installation automatic, all in one

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

`ofc_hypermesh_install.m`

or get it on the dedicated web page. Thereafter, one run it under Octave. This command download, extract and configure the *fc-hypermesh* and the required *fc-tools* package in the current directory.

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

```
# mkdir -p ~/Octave/packages
# cd ~/Octave/packages
# wget http://www.math.univ-paris13.fr/~cuvelier/software/codes/fc-
hypermesh/0.0.2/ofc_hypermesh_install.m
```

Then in a Octave terminal run the following commands

```
>> cd ~/Octave/toolboxes
>> ofc_hypermesh_install
```

There is the output of the `ofc_hypermesh_install` command:

```
1- Downloading and extracting the packages
  -> <fc-tools>[0.0.18] ... OK
  -> <fc-hypermesh>[0.0.4] ... OK
2- Setting the <fc-hypermesh> package
Write in ~/Octave/packages/fc-hypermesh-full/ofc-hypermesh-0.0.4/
  configure_loc.m ...
  -> done
3- Using the <fc-hypermesh> package
Under Octave:
  addpath('~/Octave/packages/fc-hypermesh-full/ofc-hypermesh-0.0.4')
  fc_hypermesh.init()

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

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

```
>> addpath('~/Octave/packages/fc-hypermesh-full/ofc-hypermesh-0.0.4')
>> fc_hypermesh.init()
```

For **uninstalling**, one just have to delete directory `~/Octave/packages/fc-hypermesh-full`

3 Using the fc-hypermesh package

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

3.1 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. An elementary mesh class object `EltMesh` is used to store only one mesh, the main mesh as well as any of the m -face meshes. This class `EltMesh` also simplify (for me) the codes writing. Its fields are the following:

- d , space dimension
- m , kind of mesh ($m = d$ for the main mesh)
- type, 0 for simplicial mesh or 1 for orthotope mesh
- n_q , number of vertices
- q , vertices array of dimension d -by- n_q
- n_{me} , number of mesh elements
- me , connectivity array of dimension $(d + 1)$ -by- n_{me} for simplices elements or 2^d -by- n_{me} 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)

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 its m -face meshes, $0 \leq m < d$. Its Fields are the following

- d : space dimension
- type: string 'simplicial' or 'orthotope' mesh
- Mesh: main mesh as an `EltMesh` object
- Faces: list of arrays of `EltMesh` objects such that `Faces(1)` is an array of all the $(d - 1)$ -face meshes, `Faces(2)` is an array of all the $(d - 2)$ -face meshes, and so on
- box: a d -by-2 array such that $\text{box}(i, 1) = a_i$ and $\text{box}(i, 2) = b_i$.

3.1.1 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:

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

Options are defined with one or more Name,Value pair arguments. The Name argument could be the string

- 'box' : used to specify the d-orthotope $[a_1, b_1] \times \dots \times [a_d, b_d]$ by setting Value as an d-by-2 array such that $a_i = \text{Value}(i,1)$ and $b_i = \text{Value}(i,2)$.
- 'type' : used to select the kind of elements used for meshing. The default value is 'simplicial' and otherwise 'orthotope' can be used.

3.1.2 plotmesh method

The plotmesh() member function can be used to represent the mesh given by an OrthMesh object if the space dimension is leather or equal to 3. Some options are proposed with this function:

plotmesh(Name,Value)

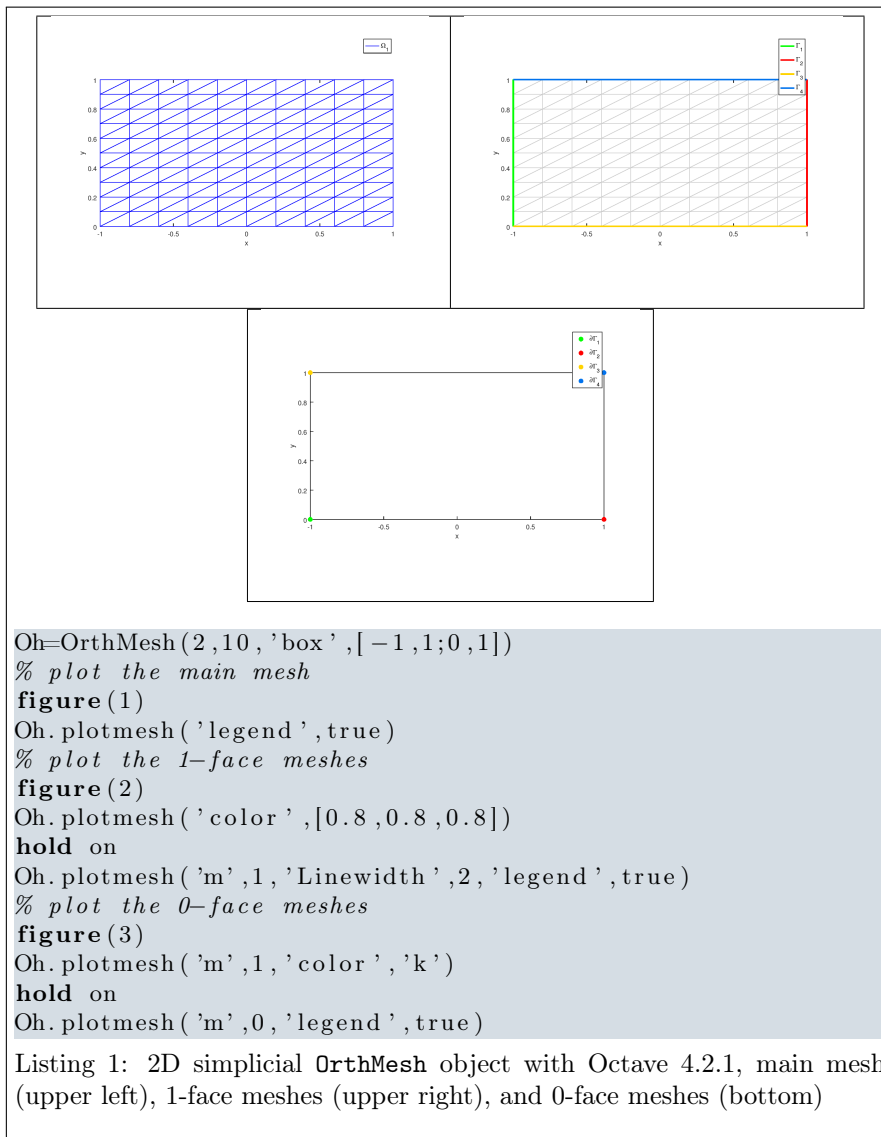
- 'legend=_value' : if value is True, a legend is displayed. Default is False.
- m = value : plots all the m-faces of the mesh. Default m = d i.e. the main mesh. ($0 \leq m \leq d$)
- ...

3.2 2d-orthotope meshing by simplices

In Listing 15, an OrthMesh object is built under Octave by using command

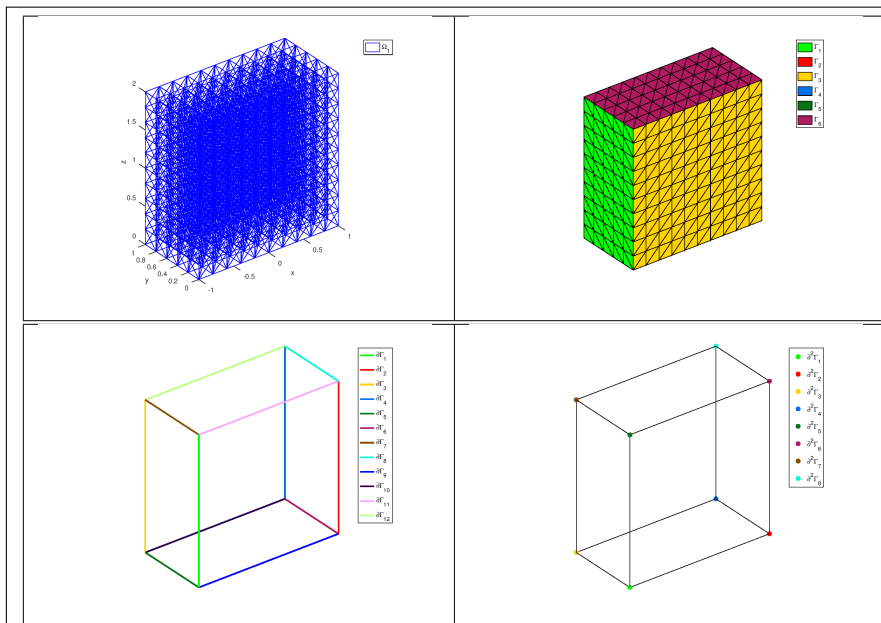
```
Oh=OrthMesh(2,10,'box',[-1,1;0,1])
```

So the Oh object is the tessellations of the orthotope $[-1, 1] \times [0, 1]$ with simplicial elements. In each direction $10 + 1 (= 11!)$ points are taken. So we have 11^2 vertices in this mesh. The main mesh and all the m -face meshes of the resulting object are plotted by using plotmesh method.



3.3 3d-orthotope meshing by simplices

In Listing 22, 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.



```

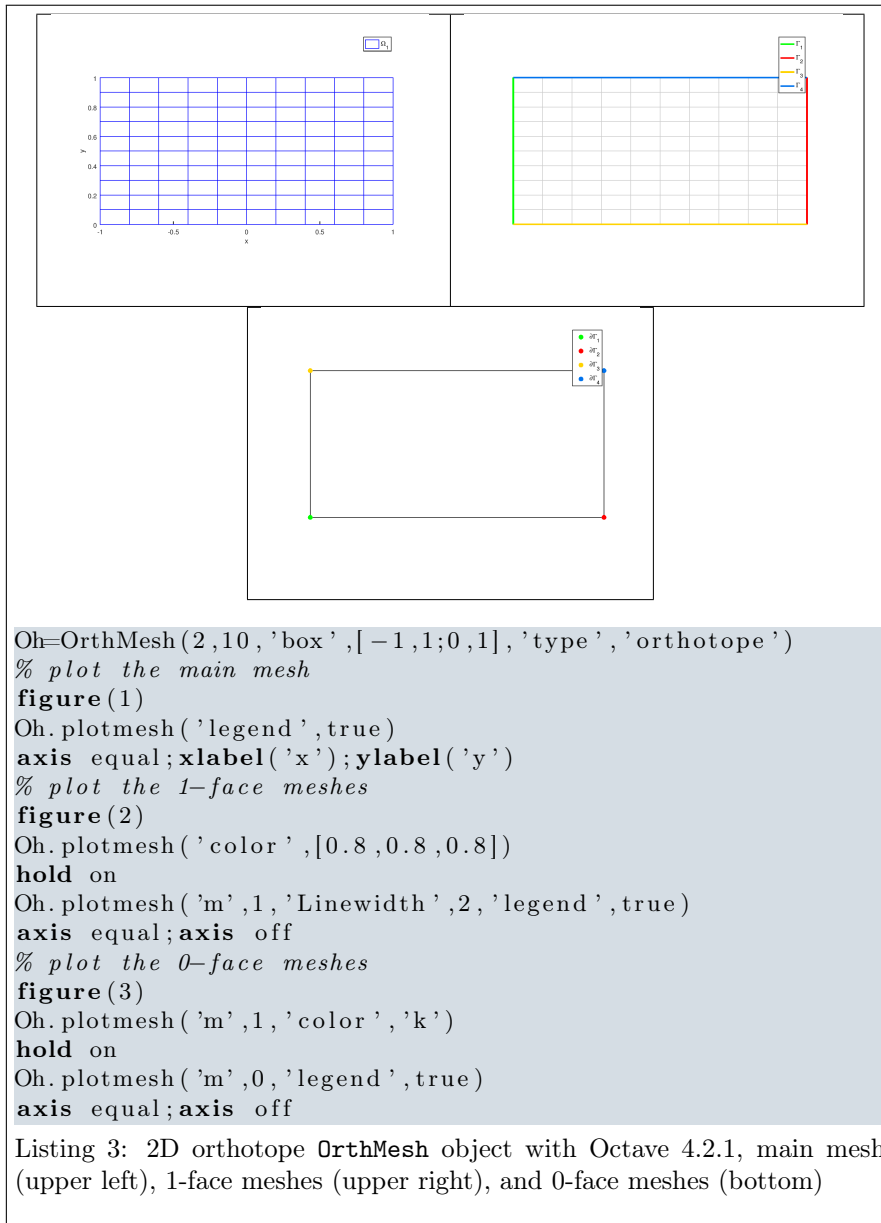
Oh=OrthMesh(3,[10,5,10], 'box', [-1,1;0,1;0,2])
% plot the main mesh
figure(1)
Oh.plotmesh('legend', true)
axis equal; xlabel('x'); ylabel('y'); zlabel('z')
% plot the 2-face meshes
figure(2)
Oh.plotmesh('m', 2, 'legend', true)
axis equal; axis off
% plot the 1-face meshes
figure(3)
Oh.plotmesh('m', 2, 'color', [0.8,0.8,0.8], 'EdgeAlpha', 0.2, ...
'FaceColor', 'none')
hold on
Oh.plotmesh('m', 1, 'Linewidth', 2, 'legend', true)
axis equal; axis off
% plot the 0-face meshes
figure(4)
Oh.plotmesh('m', 1, 'color', 'k')
hold on
Oh.plotmesh('m', 0, 'legend', true)
axis equal; axis off

```

Listing 2: 3D simplicial `OrthMesh` object with Octave 4.2.1, main mesh (upper left), 2-face meshes (upper right), 1-face meshes (bottom left) and 0-face meshes (bottom right)

3.4 2d-orthotope meshing by orthotopes

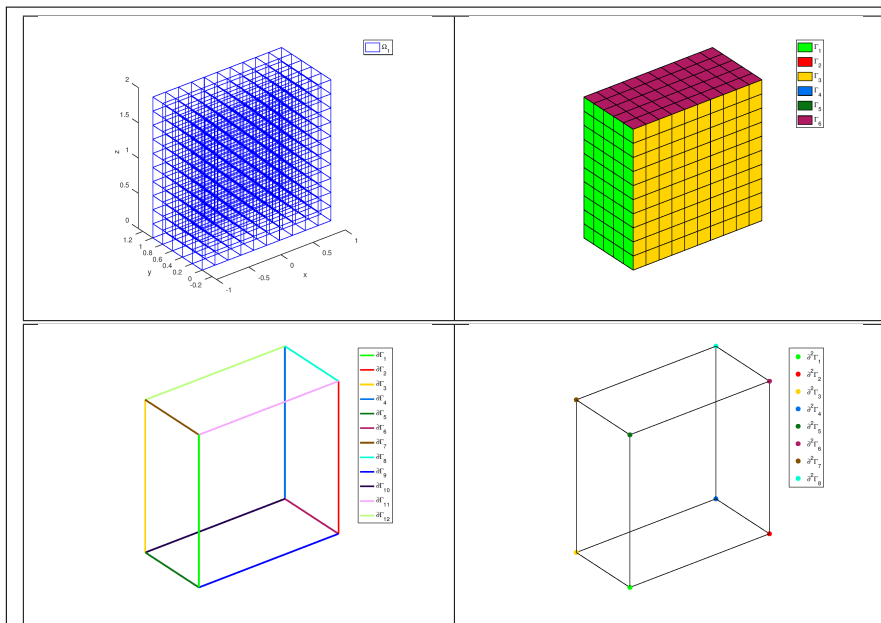
In Listing 22, an `OrthMesh` object is built under Octave for the orthotope $[-1, 1] \times [0, 1] \times [0, 2]$ with orthotope elements and $\mathbf{N} = (10, 5, 10)$. The main mesh and all the m -face meshes of the resulting object are plotted.



3.5 3d-orthotope meshing by orthotopes

In Listing 22, 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 m -face meshes of the resulting object are plotted.



```

Oh=OrthMesh(3,[10,5,10], 'box', [-1,1;0,1;0,2], ...
    'type', 'orthotope')
% plot the main mesh
figure(1)
Oh.plotmesh('legend', true)
axis equal; xlabel('x'); ylabel('y'); zlabel('z')
% plot the 2-face meshes
figure(2)
Oh.plotmesh('m', 2, 'legend', true)
axis equal; axis off
% plot the 1-face meshes
figure(3)
Oh.plotmesh('m', 2, 'color', [0.8, 0.8, 0.8], 'EdgeAlpha', 0.2, ...
    'FaceColor', 'none')
hold on
Oh.plotmesh('m', 1, 'Linewidth', 2, 'legend', true)
axis equal; axis off
% plot the 0-face meshes
figure(4)
Oh.plotmesh('m', 1, 'color', 'k')
hold on
Oh.plotmesh('m', 0, 'legend', true)
axis equal; axis off

```

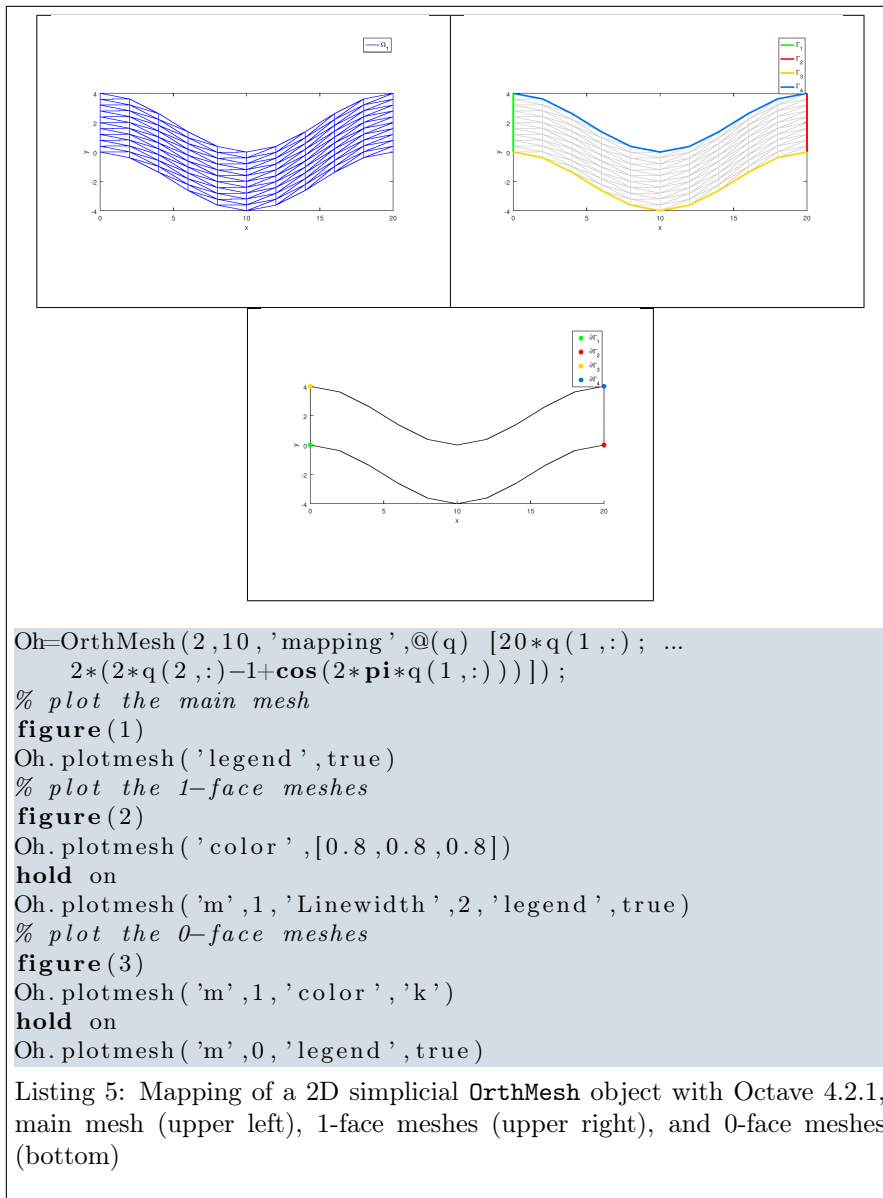
Listing 4: 3D orthotope `OrthMesh` object with Octave 4.2.1, main mesh (upper left), 2-face meshes (upper right), 1-face meshes (bottom left) and 0-face meshes (bottom right)

3.6 Mapping of a 2d-orthotope meshing by simplices

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

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

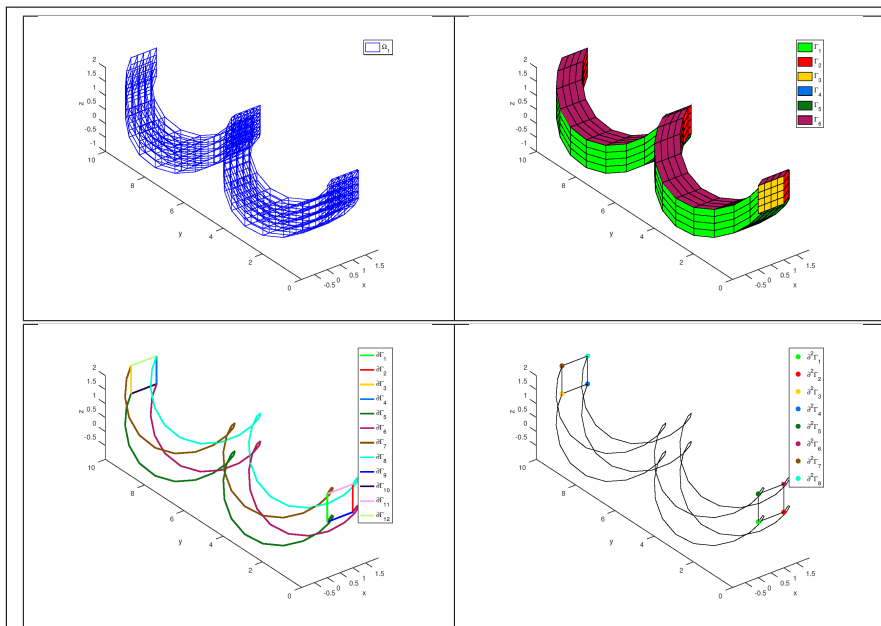
$$\begin{pmatrix} x \\ y \end{pmatrix} \longrightarrow F(x, y) = \begin{pmatrix} 20x \\ 2(2y - 1 + \cos(2\pi x)) \end{pmatrix}$$



3.7 3d-orthotope meshing by orthotopes

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

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



```

Map=@(q) [q(1,:) + sin(4*pi*q(2,:)); 10*q(2,:); ...
          q(3,:) + cos(4*pi*q(2,:))];
Oh=OrthMesh(3,[4,25,4], 'mapping',Map, 'type', 'orthotope');
% plot the main mesh
figure(1)
Oh.plotmesh()
legend('show')
% plot the 2-face meshes
figure(2)
Oh.plotmesh('m',2)
legend('show')
% plot the 1-face meshes
figure(3)
Oh.plotmesh('m',2, 'color',[0.8,0.8,0.8], 'EdgeAlpha',0.2, ...
           'FaceColor','none')
hold on
Oh.plotmesh('m',1, 'Linewidth',2, 'legend',true)
% plot the 0-face meshes
figure(4)
Oh.plotmesh('m',1, 'color','k')
hold on
Oh.plotmesh('m',0, 'legend',true)

```

Listing 6: Mapping of a 3D orthotope `OrthMesh` object with Octave 4.2.1, main mesh (upper left), 2-face meshes (upper right), 1-face meshes (bottom left) and 0-face meshes (bottom right)