



FC_SIMESH package, User's Guide *

François Cuvelier[†]

June 23, 2017

Abstract

This object-oriented Python package allows to use simplices meshes generated from `gmsh` (in dimension 2 or 3) or an hypercube triangulation (in any dimension).

Contents

1	Introduction	2
1.1	Prerequisite	2
1.2	Installation	2
1.3	Remarks	2
2	Mesh Objects	2
2.1	<code>siMESH</code> ELT object	3
2.2	<code>siMESH</code> object	3
2.3	Mesh samples	4
2.3.1	2-simplicial mesh in \mathbb{R}^2	4
2.3.2	Sample of a 3-simplicial mesh in \mathbb{R}^3	5
2.3.3	Sample of a 2-simplicial mesh in \mathbb{R}^3	6
2.4	<code>siMESH</code> object methods	7
2.4.1	<code>siMESH</code> constructor	7
2.4.2	<code>find</code> method	8
2.4.3	<code>feval</code> method	8
2.5	Hypercube as a <code>siMESH</code> object	10

*Compiled with Python 3.6.0, packages `FC_HYPERMESH-0.0.5`, `FC_OOGMSH-0.0.4`, `FC_TOOLS-0.0.13`, and the plotting libraries `MATPLOTLIB-2.0.0`, `MAYAVI-4.5.0`

[†]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 **experimental** `fc_simesh` package does not contain graphical tools natively. However two add-ons were written to integrate graphical tools : the `fc_simesh_mayavi` package [?] and the `fc_simesh_matplotlib` [?] package which use respectively Mayavi ($\geq 4.5.0$) and matplotlib ($\geq 2.0.0$) packages. Installation of Mayavi package for Python 3.6.0 under the linux distributions CentOS 7 and Ubuntu 14.04 LTS is presented on my web page <http://www.math.univ-paris13.fr/~cuvelier> in the section *Informatique/Python*.

1.1 Prerequisite

This package needs the `fc_tools`, `fc_oogmsh` and `fc_hypermesh` Python packages.

1.2 Installation

See <http://www.math.univ-paris13.fr/~cuvelier/software/fc-simesh-Python.html>

1.3 Remarks

In all this report, for graphical representation, we use the `fc_simesh_matplotlib` package with `siplt` namespace for graphical functions using matplotlib and the `fc_simesh_mayavi` package with `simlab` namespace for graphical functions using matplotlib. There is the python code previously used before some of the listings given thereafter:

```
from fc_oogmsh import gmsh
from fc_simesh.siMesh import siMesh, HyperCube
# Graphical packages :
import matplotlib.pyplot as plt
import fc_simesh_matplotlib.siMesh as siplt
from mayavi import mlab
import fc_simesh_mayavi.siMesh as simlab
```

2 Mesh Objects

In geometry, a simplex is a generalization of the notion of a triangle or tetrahedron to arbitrary dimensions. Specifically, a k -simplex in \mathbb{R}^{\dim} , $k \leq \dim$, is a polytope which is the convex hull of its $k + 1$ vertices of \mathbb{R}^{\dim} . More formally, suppose the $k + 1$ vertices $q^0, \dots, q^k \in \mathbb{R}^{\dim}$ such that $q^1 - q^0, \dots, q^k - q^0$ are linearly independent. Then, the k -simplex K determined by them is the set of points

$$K = \left\{ \sum_{i=0}^k \lambda_i q^i \mid \lambda_i \geq 0, i \in \llbracket 0, k \rrbracket, \text{ with } \sum_{i=0}^k \lambda_i = 1 \right\}.$$

We denote by k -**simplicial elementary mesh** in \mathbb{R}^{\dim} , $k \leq \dim$, a mesh with **unique label** only composed with k -simplices.

A d -**simplicial mesh** in \mathbb{R}^{\dim} , $d \leq \dim$, is an union of k -simplicial elementary meshes with $k \in \llbracket 0, d \rrbracket$.

2.1 `siMESH` object

An elementary d -simplicial mesh in dimension \dim is represented by the class `siMESH`. We give properties of this class :

Properties of `siMESH` object for d -simplicial elementary meshes in \mathbb{R}^{\dim}

<code>dim</code>	: integer space dimension
<code>d</code>	: integer ($0 \leq d \leq \dim$)
<code>n_q</code>	: integer number of vertices
<code>n_{me}</code>	: integer number of elements (d -simplices)
<code>q</code>	: <code>n_q</code> -by- <code>dim</code> numpy array of reals array of vertex coordinates
<code>me</code>	: $(d + 1)$ -by- <code>n_{me}</code> numpy array of integers connectivity array for mesh elements
<code>vols</code>	: <code>n_{me}</code> numpy array of reals array of mesh element volumes
<code>h</code>	: double mesh step size (=maximum edge length in the mesh)
<code>toGlobal</code>	: <code>n_q</code> numpy array of integers convert from local to global mesh vertices numbering
<code>toParent</code>	: <code>n_q</code> array of integers convert from local to parent mesh vertices numbering (same as global if not part of a partitioned mesh)

More precisely

- $q[j, \nu]$ is the ν -th coordinate of the $(j + 1)$ -th vertex, $\nu \in \{0, \dots, \dim - 1\}$, $j \in \{0, \dots, n_q - 1\}$. The $(j + 1)$ -th vertex will be also denoted by $q^{j+1} = q[j]$.
- $me[\beta, k]$ is the storage index of the $(\beta + 1)$ -th vertex of the $k + 1$ -th element (d -simplex), in the array q , for $\beta \in \{0, \dots, d\}$ and $k \in \{0, \dots, n_{me} - 1\}$. So $q[me[\beta, k]]$ represents the coordinates of the $(\beta + 1)$ -th vertex of the $(k + 1)$ -th mesh element.
- $vols[k]$ is the volume of the $(k + 1)$ -th d -simplex .

2.2 `siMESH` object

A d -simplicial mesh in dimension \dim , represented as an `siMESH` object, is an union of `siMESH` objects which are elementary l -simplicial meshes ($l \leq d$) in space dimension \dim .



siMESH object properties

dim	: integer space dimension
d	: integer d-dimensional simplicial mesh
sTh	: list of siMESH _{ELT} objects
nsTh	: number of siMESH _{ELT} objects
sThsimp	: numpy array of nsTh integers <i>i</i> -th siMESH _{ELT} object in sTh is a sThsimp[<i>i</i> - 1]- simplicial elementary mesh
sThlab	: numpy array of nsTh integers in sTh label of <i>i</i> -th siMESH _{ELT} object in sTh is number sThlab[<i>i</i> - 1]
n _q	: integer number of vertices in the mesh
toGlobal	: numpy array of n _q integers convert from local to global mesh vertices numbering
toParent	: numpy array of n _q integers convert from local to parent mesh vertices numbering (same as global if not part of a partitioned mesh)

Let \mathcal{T}_h be a siMESH object. The global dim-by- $\mathcal{T}_h.n_q$ numpy array q of mesh vertices is not explicitly stored in \mathcal{T}_h , however one can easily build it if necessary:

$$q[\mathbf{p}^i] = \mathcal{T}_h.sTh[i].q, \quad \forall i \in \{0, \dots, \mathcal{T}_h.nsTh - 1\} \quad (1)$$

where $\mathbf{p}^i = \mathcal{T}_h.sTh[i].toParent$.

2.3 Mesh samples

2.3.1 2-simplicial mesh in \mathbb{R}^2

Listing 1: : 2D siMESH object from sample20.geo

```
# generate .msh file from sample20.geo by using gmsh
meshfile=gmsh.buildmesh2d('sample20',20,verbose=0)
# generate a siMesh object from .msh file
Th=siMesh(meshfile)
print(Th)
```

Output

```
siMesh object
  d : 2
  dim : 2
  nq : 2558
  nme : 4954
  sTh : list of 11 siMeshElt
  nsTh : 11
  sThsimp : (11,) ndarray
  [1 1 1 1 1 1 1 2 2 2 2]
  sThlab : (11,) ndarray
  [ 1  2 20 101 102 103 104 1  2 10 20]
```

For example, from output of Listing 1 or Figure 1 the complete domain is

$$\Omega = \Omega_1 \cup \Omega_2 \cup \Omega_{10} \cup \Omega_{20}$$

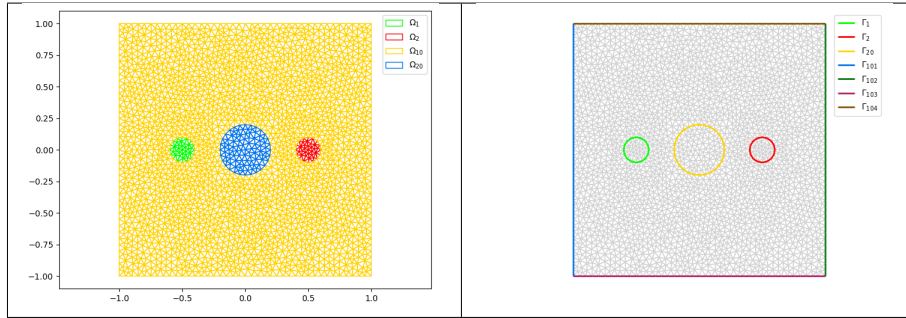


Figure 1: Mesh from `sample20.geo`, label of the domains (left) and label of the boundaries (right)

and we note

$$\Gamma = \Gamma_1 \cup \Gamma_2 \cup \Gamma_{20} \cup \Gamma_{101} \cup \Gamma_{102} \cup \Gamma_{103} \cup \Gamma_{104}.$$

So this mesh is 2-simplicial mesh in \mathbb{R}^2 and is composed of :

- four 2-simplicial elementary meshes : $\Omega_i, \forall i \in \{1, 2, 10, 20\}$
- seven 1-simplicial elementary meshes : $\Gamma_i \forall i \in \{1, 2, 20, 101, 102, 104\}$

2.3.2 Sample of a 3-simplicial mesh in \mathbb{R}^3

```

Listing 2 : 3D siMESH object from quart_sphere2.geo
meshfile=gmsh.buildmesh3d('quart_sphere2',5, verbose=0)
Th=siMesh(meshfile)
print(Th)

Output

Mesh /fcopt/python3.6.0/lib/python3.6/site-packages/meshes/quart_sphere2-5.msh is a
3-dimensional mesh
Force dimension to 3

siMesh object
  d : 3
  dim : 3
  nq : 1123
  nme : 4552
  sTh : list of 18 siMeshElt
  nsTh : 18
  sThsimp : (18,) ndarray
  [1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 3 3]
  sThlab : (18,) ndarray
  [1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 1 2]

```

The mesh obtained from Listing 2 or 2 is a 3-simplicial mesh in \mathbb{R}^3 and is composed of :

- two 3-simplicial elementary meshes : $\Omega_i, \forall i \in \{1, 2\}$
- seven 2-simplicial elementary meshes : $\Gamma_i \forall i \in \llbracket 1, 7 \rrbracket$
- nine 1-simplicial elementary meshes : $\partial\Gamma_i \forall i \in \llbracket 1, 9 \rrbracket$

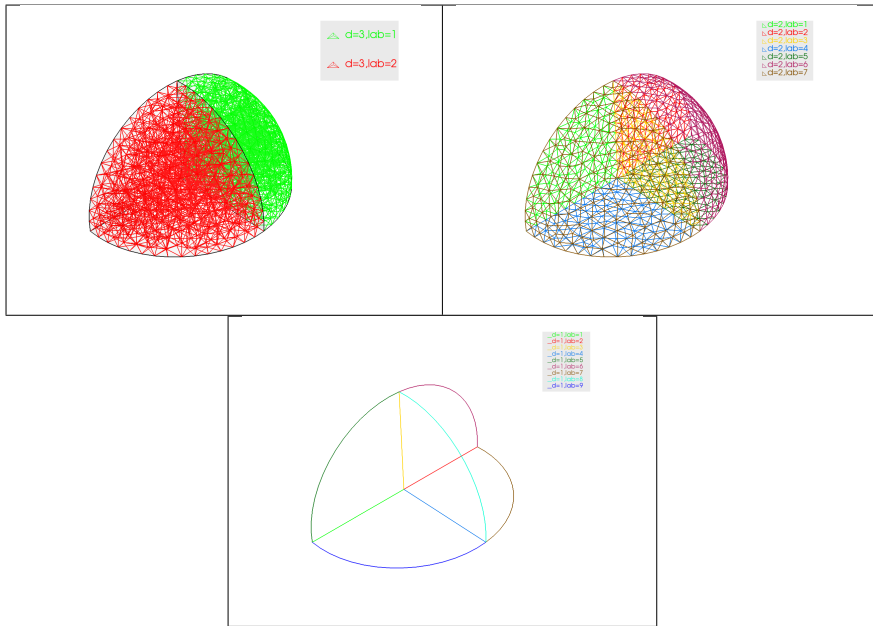


Figure 2: Mesh from `quart_sphere2.geo`, label of the domains (top left) , label of the boundaries (top right) and label of the 1-simplicial elementary meshes (bottom)

2.3.3 Sample of a 2-simplicial mesh in \mathbb{R}^3

```

Listing 3: : 3D surface siMESH object from demisphere4surf.geo
meshfile=gmshtoolkit.buildmesh3ds(' demisphere4surf ', 5, verbose=0)
Th=siMesh(meshfile)
print(Th)

Output

Mesh /fcopt/python3.6.0/lib/python3.6/site-packages/meshes/demisphere4surf-5.msh is a
3-dimensional mesh
Force dimension to 3

siMesh object
  d : 2
  dim : 3
  nq : 209
  nme : 384
  sTh : list of 12 siMeshElt
  nsTh : 12
  sThsimp : (12,) ndarray
  [1 1 1 1 1 1 1 1 2 2 2 2]
  sThlab : (12,) ndarray
  [1 2 3 4 5 6 7 8 1 2 3 4]

```

The mesh obtained from Listing 3 or 3 is a 2-simplicial mesh in \mathbb{R}^3 and is composed of :

- four 2-simplicial elementary meshes : $\Omega_i, \forall i \in \llbracket 1, 4 \rrbracket$
- eight 1-simplicial elementary meshes : $\Gamma_i \forall i \in \llbracket 1, 8 \rrbracket$

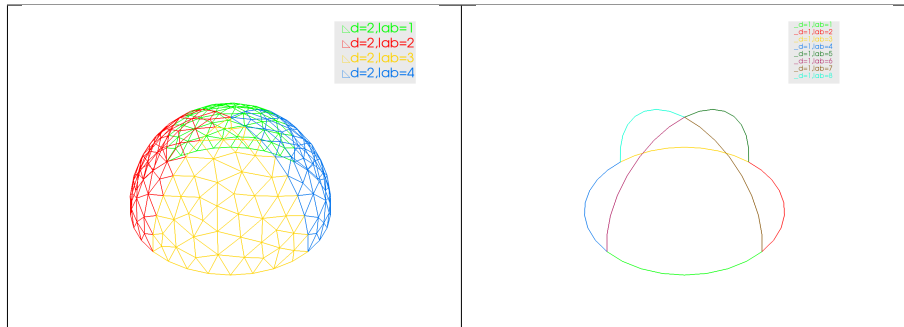


Figure 3: Mesh from `sample20.geo`, label of the domains (left) and label of the boundaries (right)

2.4 siMESH object methods

2.4.1 siMESH constructor

The constructor of the `siMESH` class can initialize the object from a `gmsh` mesh file saved as a `.msh` file

Syntaxe

```
Th=siMesh(meshfile)
Th=siMesh(meshfile,Name=Value)
```

Description

`Th=siMesh(meshfile)` create the `siMESH` object `Th` from the mesh file `meshfile` (`gmsh` format by default).

`Th=siMesh(meshfile,Name=Value, ...)` specifies function options using one or more `Name,Value` pair arguments. The `Name` options can be

- `dim` : to specify the space dimension (default 2),
- `d` : to specify the dimensions of the simplices to read, (default `[dim,dim-1]`)

Examples The following example use the function `gmsh.buildmesh2d` of the `fc_oogmsh` package to build the mesh from the `.geo` file `condenser11.geo`. This `.geo` file is located in the directory `geodir` of the `fc_oogmsh` package.

Listing 4: : siMESH constructor sample

```
meshfile=gmshtool.buildmesh2d('condenser11',400,force=True,verbose=0)
Th=siMesh(meshfile)
print(Th)
```

Output

```
siMesh object
  d : 2
  dim : 2
  nq : 823105
  nme : 1642008
  sTh : list of 19 siMeshElt
  nsTh : 19
  sThsimp : (19,) ndarray
  [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
  sThlab : (19,) ndarray
  [ 1  2  3  4  5  6  7  8 20 101 102 103 104 2  4  6  8 10
    20]
```

2.4.2 find method

We denote by Th a siMESH object.

- `Th.find(d)` : returns the indices numpy array of the d-simplicial elementary meshes in the array Th.sTh.
- `Th.find(d, labels=value)` : returns the indices of the d-simplicial elementary meshes with label in value. value could be an integer, a list of integers or a numpy array of integers. If nothing is found then return None otherwise returns the returned indices are of the same type as the input optional value.

Several examples are given in functions siMesh.demos.find2D01(), siMesh.demos.find3D01(), ... We present now some very basic samples.

2.4.3 feval method

Evaluates a function at vertices of the mesh. We denote by Th a siMESH object.

- `res=Th.feval(fun)` : the input parameter fun is either a function or a list of functions for vector-valued functions. If fun is a function then the output is a numpy array with shape (Th.nq). If fun is a list of functions then the output is a numpy array with shape (len(fun),Th.nq).
- `res=Th.feval(fun,key=value,...)` specifies function options using one or more key,value pair arguments. Key options could be
 - `d` : to specify the d-simplicial elementary meshes on which to evaluate the function (default Th.d). A zero value is set on all vertices not in these meshes.
 - `labels` : to specify the labels of the elementary meshes on which to evaluate the function (default is all). A zero value is set on all vertices not in these meshes.

Several examples are given in functions siMesh.demos.feval2D01(), siMesh.demos.feval3D01(), ... We present now some very basic samples.

Sample 1 Let $g : \mathbb{R}^2 \mapsto \mathbb{R}$ defined by $g(x, y) = \cos(x)\sin(y)$. We propose in Listing 5 four approaches to defined this function for using with feval method.

Listing 5: : feval method, four ways to defined a function

```
meshfile=gmesh.buildmesh2d('condenser11',50,verbose=0)
Th=siMesh(meshfile)

import numpy as np
g1=lambda x,y: np.cos(x)*np.sin(y)
g2=lambda X: np.cos(X[0])*np.sin(X[1])

def g3(x,y):
    return np.cos(x)*np.sin(y)

def g4(X):
    return np.cos(X[0])*np.sin(X[1])

z1=Th.feval(g1)
z2=Th.feval(g2)
z3=Th.feval(g3)
z4=Th.feval(g4)

print('max(abs(z2-z1))=%e'%max(abs(z2-z1)))
print('max(abs(z3-z1))=%e'%max(abs(z3-z1)))
print('max(abs(z4-z1))=%e'%max(abs(z4-z1)))
```

Output

```
max(abs(z2-z1))=0.000000e+00
max(abs(z3-z1))=0.000000e+00
max(abs(z4-z1))=0.000000e+00
```

Sample 2

Listing 6: : feval method, functions with parameters

```
meshfile=gmesh.buildmesh2d('condenser11',50,verbose=0)
Th=siMesh(meshfile)

import numpy as np
f=lambda x,y: np.cos(2*x)*np.sin(3*y)

def g(x,y,cx,cy):
    return np.cos(cx*x)*np.sin(cy*y)

g23=lambda x,y: g(x,y,2,3)

z=Th.feval(f)
z23=Th.feval(g23)
print('max(abs(z23-z))=%e'%max(abs(z23-z)))
```

Output

```
max(abs(z23-z))=0.000000e+00
```

Sample 3

Listing 7: : feval method with a vector-valued function

```
meshfile=gmesh.buildmesh2d('condenser11',50,verbose=0)
Th=siMesh(meshfile)

# f : R^2 -> R^3
import numpy as np
f=[lambda x,y: np.cos(2*x)*np.sin(3*y),
   lambda x,y: np.cos(3*x)*np.sin(4*y),
   lambda x,y: np.cos(4*x)*np.sin(5*y)]
z=Th.feval(f)
print('nq='+str(Th.nq)+'_z.shape='+str(z.shape))
```

Output

```
nq=13255 z.shape=(3, 13255)
```

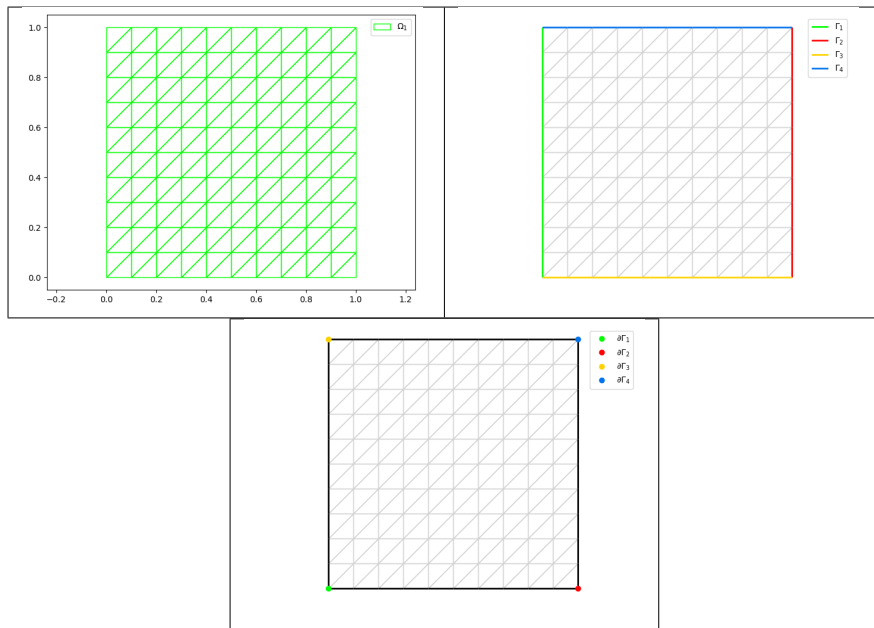


Figure 4: 2D Hypercube `siMESH` object generated with the function `siMesh.HyperCube`, representation of the elementary meshes with 2-simplices (top left), 1-simplices (top right) and 0-simplices (bottom)

2.5 Hypercube as a `siMESH` object

The function `siMesh.HyperCube` allows to create a `siMESH` object representing an hypercube in any dimension. It uses the Python package `fc_hypermesh`.

```

Listing 8: : 2D Hypercube siMESH object generated with the function siMesh.HyperCube
Th=HyperCube(2,10)
print(Th)

Output

siMesh object
  d : 2
  dim : 2
  nq : 121
  nme : 200
  sTh : list of 9 siMeshElt
  nsTh : 9
  sThsimp : (9,) ndarray
           [2 1 1 1 1 0 0 0 0]
  sThlab : (9,) ndarray
           [1 1 2 3 4 1 2 3 4]

```

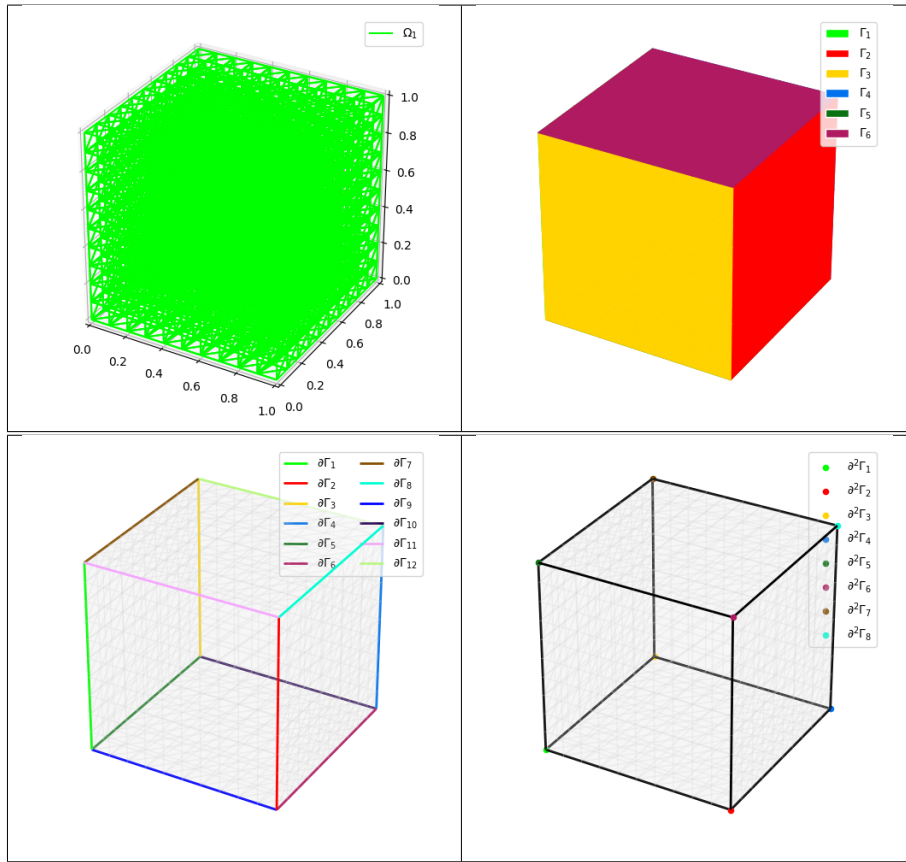


Figure 5: 3D Hypercube `siMESH` object generated with the function `siMesh.HyperCube`, representation of the elementary meshes with 3-simplices (top left), 2-simplices (top right), 1-simplices (bottom left) and 0-simplices (bottom right)

```

Listing 9: : 3D Hypercube siMESH object generated with the function siMesh.HyperCube
Th=HyperCube(3,10)
print(Th)

Output

siMesh object
  d : 3
  dim : 3
  nq : 1331
  nme : 6000
  sTh : list of 27 siMeshElt
  nsTh : 27
sThsimp : (27,) ndarray
[3 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0]
sThlab : (27,) ndarray
[ 1 1 1 2 3 4 5 6 1 2 3 4 5 6 7 8 9 10 11 12 1 2 3 4 5 6
 7 8]

```

Listing 10: : function siMesh.HyperCube

```
Th=HyperCube(4,10)
print(Th)
```

Output

```
siMesh object
  d : 4
  dim : 4
  nq : 14641
  nme : 240000
  sTh : list of 81 siMeshElt
  nsTh : 81
  sThsimp : (81,) ndarray
  [4 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1
   1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
   0 0 0 0 0 0 0]
  sThlab : (81,) ndarray
  [ 1 1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
   17 18 19 20 21 22 23 24 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
   18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 1 2 3 4 5 6 7 8 9 10
   11 12 13 14 15 16]
```

Listing 11: : function siMesh.HyperCube

```
Th=HyperCube(5,6)
print(Th)
```

Output

```
siMesh object
  d : 5
  dim : 5
  nq : 16807
  nme : 933120
  sTh : list of 243 siMeshElt
  nsTh : 243
  sThsimp : (243,) ndarray
  [5 4 4 4 4 4 4 4 4 4 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
   3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
   2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
   2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
   1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
   1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   0]
  sThlab : (243,) ndarray
  [ 1 1 2 3 4 5 6 7 8 9 10 1 2 3 4 5 6 7 8 9 10 11 12 13 14
   15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
   40 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
   25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
   50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
   75 76 77 78 79 80 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
   20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44
   45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69
   70 71 72 73 74 75 76 77 78 79 80 1 2 3 4 5 6 7 8 9 10 11 12 13 14
   15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32]
```