



User's Guide *

François Cuvelier[†]

December 13, 2017

Abstract

This object-oriented Matlab toolbox allows to use simplices meshes generated from `gmsh` (in dimension 2 or 3) or an hypercube triangulation (in any dimension). For graphical representation the `FC-SIPLT` toolbox is used.

0 Contents

1	Introduction	2
2	Installation	2
2.1	Installation automatic, all in one (recommanded)	2
3	Mesh Objects	4
3.1	<code>siMESH</code> ELT object	4
3.2	<code>siMESH</code> object	6
3.3	Mesh samples	7
3.3.1	2-simplicial mesh in \mathbb{R}^2	7
3.3.2	Sample of a 3-simplicial mesh in \mathbb{R}^3	9
3.3.3	Sample of a 2-simplicial mesh in \mathbb{R}^3	11
3.4	<code>siMESH</code> object methods	12
3.4.1	<code>siMESH</code> constructor	12
3.4.2	<code>find</code> method	13
3.4.3	<code>feval</code> method	14

*Compiled with Matlab 2017a

[†]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.

3.4.4	eval method	15
3.4.5	get_h method	17
3.4.6	get_mesh method	17
3.4.7	get_nme method	17
3.4.8	get_nq method	18
3.5	Hypercube as a siMESH object	19
3.5.1	2D hypercube	19
3.5.2	3D hypercube	20
3.5.3	4D hypercube	21
3.5.4	5D hypercube	22

1 Introduction

This toolbox was tested under

Windows 10.0.16299: with Matlab R2015b to R2017b

Mac OS X 10.12.6: with Matlab R2015b to R2017b

Ubuntu 14.04.5 LTS: with Matlab R2015b to R2017b

Ubuntu 16.04.3 LTS: with Matlab R2015b to R2017b

Ubuntu 17.10: with Matlab R2015b to R2017b

2 Installation

2.1 Installation automatic, all in one (recommanded)

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

```
mfc_simesh_install.m
```

or get it on the dedicated web page. Thereafter, one run it under Matlab. This command download, extract and configure the *fc-simesh* and the required toolboxes (*fc-tools*, *fc-oogmsh*, *fc-hypermesh*) in the current directory.

For example, to install this toolbox in `~/Matlab/toolboxes` directory, one have to copy the file `mfc_simesh_install.m` in the `~/Matlab/toolboxes` directory. Then in a Matlab terminal run the following commands

```
>> cd ~/Matlab/toolboxes
>> mfc_simesh_install
```

There is the output of the `mfc_simesh_install` command on a Linux computer:

```

Parts of the Matlab <fc-simesh> toolbox.
Copyright (C) 2016-2017 F. Cuvelier <cuvelier@math.univ-paris13.fr>

1- Downloading and extracting the toolboxes
-> <fc-tools>[0.0.19] ... OK
-> <fc-hypermesh>[0.0.6] ... OK
-> <fc-oogmsh>[0.0.17] ... OK
-> <fc-simesh>[0.2.1] ... OK
-> <fc-graphics4mesh>[0.0.2] ... OK
-> <fc-siplt>[0.0.2] ... OK
2- Setting the toolboxes
2-a) Setting the <fc-hypermesh> toolbox
Write in ...
~/Matlab/toolboxes/fc-simesh-full/fc_hypermesh-0.0.6/configure_loc.m ...
...
-> done
2-b) Setting the <fc-oogmsh> toolbox
[fc-oogmsh] Using GMSH binary : ~/bin/gmsh
[fc-oogmsh] Writing in ...
~/Matlab/toolboxes/fc-simesh-full/fc_oogmsh-0.0.17/configure_loc.m ...
...
[fc-oogmsh] configured with
-> gmsh_bin='~/bin/gmsh';
-> ...
-> mesh_dir='~/Matlab/toolboxes/fc-simesh-full/fc_oogmsh-0.0.17/meshes';
-> ...
-> geo_dir='~/Matlab/toolboxes/fc-simesh-full/fc_oogmsh-0.0.17/geodir';
-> fc_tools_dir='~/Matlab/toolboxes/fc-simesh-full/fc_tools-0.0.19';
[fc-oogmsh] done
2-c) Setting the <fc-simesh> toolbox without graphics
2-d) Setting the <fc-graphics4mesh> toolbox
Write in ...
~/Matlab/toolboxes/fc-simesh-full/fc_graphics4mesh-0.0.2/configure_loc.m ...
...
-> done
2-e) Setting the <fc-siplt> toolbox
Write in ...
~/Matlab/toolboxes/fc-simesh-full/fc_siplt-0.0.2/configure_loc.m ...
-> done
2-f) Setting the <fc-simesh> toolbox with graphics
[fc-simesh] Writing in ...
~/Matlab/toolboxes/fc-simesh-full/fc_simesh-0.2.1/configure_loc.m ...
[fc-simesh] configured with
-> oogmsh_dir ...
= '~/Matlab/toolboxes/fc-simesh-full/fc_oogmsh-0.0.17';
-> hypermesh_dir ...
= '~/Matlab/toolboxes/fc-simesh-full/fc_hypermesh-0.0.6';
-> siplt_dir = '~/Matlab/toolboxes/fc-simesh-full/fc_siplt-0.0.2';
[fc-simesh] done
3- Using instructions
To use the <fc-simesh> toolbox:
addpath('~/Matlab/toolboxes/fc-simesh-full/fc_simesh-0.2.1')
fc_simesh.init()

See ~/Matlab/toolboxes/mfc_simesh_set.m

```

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

```

>> addpath('~/Matlab/toolboxes/fc-simesh-full/fc-simesh-0.2.1')
>> fc_simesh.init()

```

To install the *fc-simesh* package without graphical extension one can use the following command

```

>> mfc_simesh_install('graphics',false)

```

For **uninstalling**, one just have to delete directory

`~/Matlab/toolboxes/fc-simesh-full`

3 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 $k \in \llbracket 0, d \rrbracket$.

3.1 `SIMESH` object

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



Properties of `siMESHelt` 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>	: dim-by- <code>n_q</code> array of reals array of vertex coordinates
<code>me</code>	: (d + 1)-by- <code>n_{me}</code> array of integers connectivity array for mesh elements
<code>vols</code>	: 1-by- <code>n_{me}</code> array of reals array of mesh element volumes
<code>h</code>	: double mesh step size (=maximum edge length in the mesh)
<code>toGlobal</code>	: 1-by- <code>n_q</code> array of integers convert from local to global mesh vertices numbering. Prefer the use of <code>toParents{end}</code> instead. <i>It will be removed in a future release.</i>
<code>toParent</code>	: 1-by- <code>n_q</code> array of integers convert from local to parent mesh vertices numbering (same as <code>toGlobal</code> if not part of a partitioned mesh). Prefer the use of <code>toParents{1}</code> instead. <i>It will be removed in a future release.</i>
<code>nqParents</code>	: 1-by- <code>n</code> array of integers <code>nqParents(1)</code> number of vertices in the <i>parent</i> mesh, <code>nqParents(2)</code> number of vertices in the <i>parent</i> of the <i>parent</i> mesh, <code>nqParents(end)</code> number of vertices in the global mesh.
<code>toParents</code>	: 1-by- <code>n</code> cell array <code>toParents{1}</code> indices array which convert local vertices numbering to the <i>parent</i> mesh vertices numbering, <code>nqParents{2}</code> indices array which convert local vertices numbering to the <i>parent</i> of the <i>parent</i> mesh, <code>nqParents{end}</code> indices array which convert local vertices numbering to the global mesh.


More precisely

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

- $\text{vols}(k)$ is the volume of the k -th d -simplex .

3.2 `siMESH` object

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

 <code>siMESH</code> object properties	
<code>dim</code>	: integer space dimension
<code>d</code>	: integer d-dimensional simplicial mesh
<code>sTh</code>	: array of <code>siMESH</code> <code>ELT</code> objects
<code>nsTh</code>	: number of <code>siMESH</code> <code>ELT</code> objects
<code>sThsimp</code>	: array of <code>nsTh</code> integers i -th <code>siMESH</code> <code>ELT</code> object in <code>sTh</code> is a <code>sThsimp(i)</code> -simplicial elementary mesh
<code>sThlab</code>	: array of <code>nsTh</code> integers in <code>sTh</code> label of i -th <code>siMESH</code> <code>ELT</code> object in <code>sTh</code> is number <code>sThlab(i)</code>
<code>nq</code>	: integer number of vertices in the mesh
<code>toGlobal</code>	: 1-by- <code>nq</code> array of integers convert from local to global mesh vertices numbering. Prefer the use of <code>toParents{end}</code> instead. <i>It will be removed in a future release.</i>
<code>toParent</code>	: 1-by- <code>nq</code> array of integers convert from local to parent mesh vertices numbering (same as <code>toGlobal</code> if not part of a partitioned mesh). Prefer the use of <code>toParents{1}</code> instead. <i>It will be removed in a future release.</i>
<code>nqParents</code>	: 1-by- n array of integers Only used with partitioned mesh and the <code>FC-PSIMESH</code> toolbox.
<code>toParents</code>	: 1-by- n cell array Only used with partitioned mesh and the <code>FC-PSIMESH</code> toolbox.

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

```
q=zeros(Th.dim,Th.nq);
for i=Th.find(Th.d)
    q(:,Th.sTh{i}.toParents{1})=Th.sTh{i}.q;
end
```

3.3 Mesh samples

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

```

Listing 1: : 2D siMESH object from sample20.geo
meshfile=gmsht.buildmesh2d('sample20',20,'force',false);
Th=siMesh(meshfile);
fprintf('***_Th:\n')
disp(Th)
fprintf('***_Th.sTh{9}:\n')
disp(Th.sTh{9})

Output

[fc-oogmsh] Input file : <fc-oogmsh>/geodir/2d/sample20.geo
[fc-oogmsh] Starting building mesh <fc-oogmsh>/meshes/sample20-20.msh with gmsht 3.0.6
[fc-oogmsh] Using command : gmsht -2 -setnumber N 20 <fc-oogmsh>/geodir/2d/sample20.geo -o ...
<fc-oogmsh>/meshes/sample20-20.msh
Be patient...
*** Th:
siMesh with properties:
  d: 2 double
  dim: 2 double
  sTh: (1x1 cell)
  nsTh: 11 double
  toGlobal: (1x2558 double)
  toParent: (1x2558 double)
  sThsimp: [ 1 1 1 1 1 1 1 1 2 2 2 2 ] (1x11 double)
  sThlab: [ 1 2 20 101 102 103 104 1 2 10 20 ] (1x11 double)
  sThcolors: (1x3 double)
  bbox: [ -1 1 -1 1 ] (1x4 double)
  sThgeolab: []
  sThphyslab: [ 1 2 10 20 ] (1x4 double)
  sThpartlabs: []
  nq: 2558 double
  nqParents: 2558 double
  toParents: (1x1 cell)
*** Th.sTh{9}:
siMeshElt with properties:
  d: 2 double
  dim: 2 double
  nq: 39 double
  nme: 60 double
  q: (2x39 double)
  me: (3x60 double)
  toGlobal: (1x39 double)
  nqGlobal: 2558 double
  toParent: (1x39 double)
  nqParent: 2558 double
  nqParents: 2558 double
  toParents: (1x1 cell)
  label: 2 double
  Tag: (1x28 char)
  color: [ 1 0 0 ] (1x3 double)
  vols: (1x60 double)
  gradBaCo: (60x3 double)
  geolab: (60x1 double)
  partlab: []
  bbox: [ 0.4 0.6 -0.1 0.1 ] (1x4 double)
  h: 0.051428 double

```

From the output of the Listing 1 or from the Figure 1 the complete domain is

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

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 : $\Gamma_i \forall i \in \{1, 2, 20, 101, 102, 104\}$

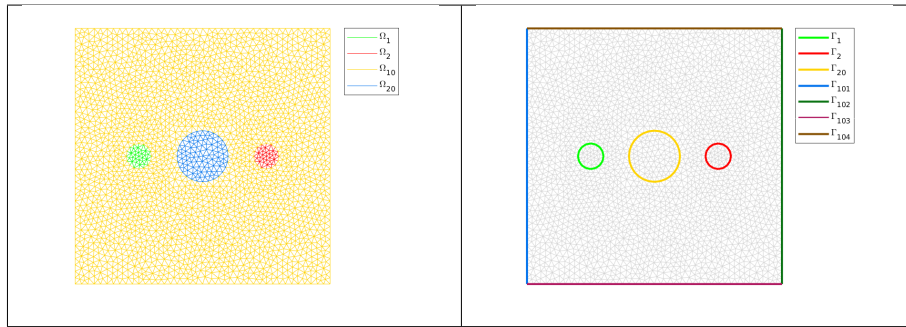


Figure 1: 2D `siMESH` object from `sample20.geo`

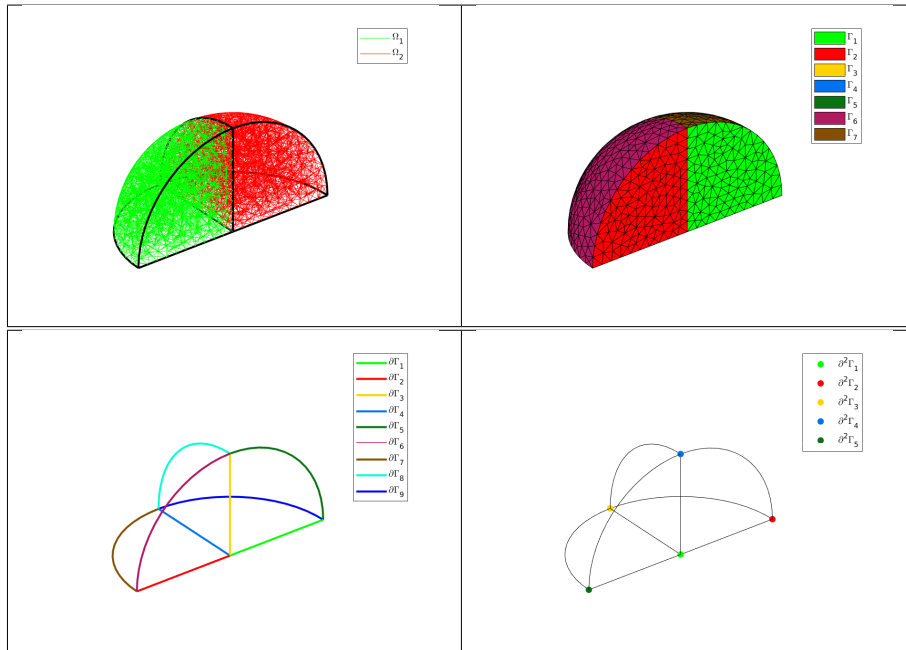


Figure 2: 3D Mesh from `quart_sphere2.geo`

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

Listing 3: 3D surface Mesh from demisphere4surf.geo

```
meshfile=gmsht.buildmesh3ds('demi sphere4surf',5,'force',true);
Th=siMesh(meshfile);
fprintf('***_Th:\n')
disp(Th)
fprintf('***_Th.sTh{9}:\n')
disp(Th.sTh{9})
```

Output

```
[fc-oogmsh] Input file : <fc-oogmsh>/geodir/3ds/demi sphere4surf.geo
[fc-oogmsh] Starting building mesh <fc-oogmsh>/meshes/demi sphere4surf-5.msh with gmsht 3.0.6
[fc-oogmsh] Using command : gmsht -2 -setnumber N 5 <fc-oogmsh>/geodir/3ds/demi sphere4surf.geo ...
-o <fc-oogmsh>/meshes/demi sphere4surf-5.msh
Be patient...
Mesh /tmp/tmp.IiKckek6YW/fc-simesh-full/fc-oogmsh-0.0.17/meshes/demi sphere4surf-5.msh is a ...
3-dimensional mesh
Force dimension to 3
*** Th:
siMesh with properties:
  d: 2 double
  dim: 3 double
  sTh: (1x12 cell)
  nsTh: 12 double
  toGlobal: (1x247 double)
  toParent: (1x247 double)
  sThsimp: [ 1 1 1 1 1 1 1 1 2 2 2 2 ] (1x12 double)
  sThlab: [ 1 2 3 4 5 6 7 8 1 2 3 4 ] (1x12 double)
  sThcolors: (1x3 double)
  bbox: [ -1 1 -1 1 0 1 ] (1x6 double)
  sThgeolab: []
  sThphyslab: [ 1 2 3 4 ] (1x4 double)
  sThpartlabs: []
  nq: 247 double
  nqParents: 247 double
  toParents: (1x1 cell)
*** Th.sTh{9}:
siMeshElt with properties:
  d: 2 double
  dim: 3 double
  nq: 70 double
  nme: 114 double
  q: (3x70 double)
  me: (3x114 double)
  toGlobal: (1x70 double)
  nqGlobal: 247 double
  toParent: (1x70 double)
  nqParent: 247 double
  nqParents: 247 double
  toParents: (1x1 cell)
  label: 1 double
  Tag: (1x29 char)
  color: [ 0 1 0 ] (1x3 double)
  vols: (1x114 double)
  gradBaCo: (114x3 double)
  geolab: (114x1 double)
  partlab: []
  bbox: (1x6 double)
  h: 0.272289 double
```

The mesh obtained from the Listing 3 or from the Figure 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 : $\Gamma_i \forall i \in \llbracket 1, 8 \rrbracket$

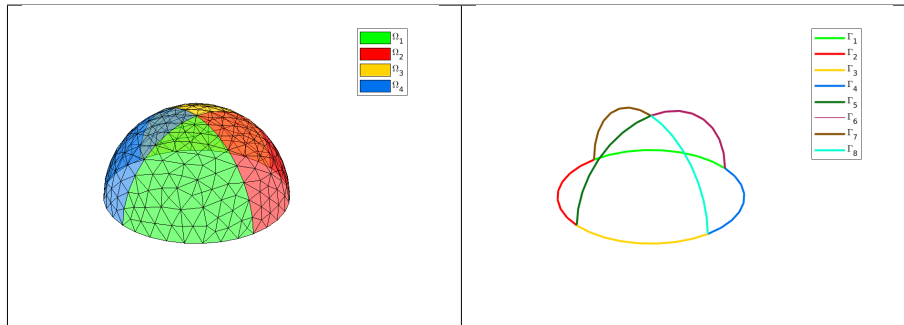


Figure 3: 3D surface Mesh from `demisphere4surf.geo`, label of the domains (left) and label of the boundaries (right)

3.4 `siMESH` object methods

`siMESH` constructor

The constructor of the `siMESH` class can initialize the object from various kind of mesh file format : `.msh` (default `gms`h format), `.mesh` (`FreeFEM++` or `Medit`) or ... (`triangle`).

Syntaxe

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

Description

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

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

- `'format'` : to specify the format of the mesh file `meshfile`. Value must be `'medit'`, `'gms`h' (default), `'freefem'` or `'triangle'`.
- `'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 `gms`h.`buildmesh2d` of the `FC-OOGMSH` toolbox to build the mesh from the `.geo` file `condenser11.geo`. This `.geo` file is located in the directory `geodir/2d` of the `FC-OOGMSH` toolbox.

Matlab commands with output

```
meshfile=gmsl.buildmesh2d('condenser11',25,'verbose',0);  
disp('***_Read_mesh_***')  
Th=siMesh(meshfile)
```

```
*** Read mesh ***  
Th =  
  
siMesh with properties:  
  d: 2 double  
  dim: 2 double  
  sTh: (1x19 cell)  
  nsTh: 19 double  
  toGlobal: (1x3474 double)  
  toParent: (1x3474 double)  
  sThsimp: [ 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 ] (1x19 double)  
  sThlab: [ 1 2 3 4 5 6 7 8 20 101 102 103 104 2 4 6 8 10 20 ] (1x19 double)  
  sThcolors: (19x3 double)  
  bbox: [ -1 1 -1 1 ] (1x4 double)  
  sThgeolab: []  
  sThphyslab: [ 2 4 6 8 10 20 ] (1x6 double)  
  sThpartlabs: []  
  nq: 3474 double  
  nqParents: 3474 double  
  toParents: (1x1 cell)
```

find method

We denote by Th a **siMESH** object.

- `Th.find(d)` : returns the sorted indices array of the d-simplicial elementary meshes in the array Th.sTh.
- `Th.find(d,labels)` : returns the sorted indices of the d-simplicial elementary meshes with label in labels. labels could be an index, an array of indices. If nothing is found then return [].

Several examples are given in functions:

`fc_simesh.demos.find2D()`, `fc_simesh.demos.find3D()`, `fc_simesh.demos.find3Ds()`

Now some very basic samples are presented.

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=fc_oogmsh.buildmesh2d('condenser11',50,'verbose',0);
Th=siMesh(meshfile);

g1=@(x,y) cos(x).*sin(y); %.* for vectorized function
g2=@(X) cos(X(1,:)).*sin(X(2,:));

z1=Th.feval(g1);
z2=Th.feval(g2);

fprintf('max(abs(z2-z1))=%e\n',max(abs(z2-z1)))

```

Output

```

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

```

Sample 2

```

Listing 6 : feval method with a vector-valued function
meshfile=fc_oogmsh.buildmesh2d('condenser11',50,'verbose',0);
Th=siMesh(meshfile)

% f : R^2 -> R^3
f=@(x,y) cos(2*x).*sin(3*y),@(x,y) cos(3*x).*sin(4*y),@(x,y) cos(4*x).*sin(5*y);
z=Th.feval(f);
fprintf('***nq=%d, size(z)=[%d,%d]',Th.nq,size(z))

```

Output

```

Th =

siMesh with properties:
    d: 2 double
   dim: 2 double
    sTh: (1x19 cell)
   nsTh: 19 double
 toGlobal: (1x13258 double)
 toParent: (1x13258 double)
 sThsimp: [ 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 ] (1x19 double)
  sThlab: [ 1 2 3 4 5 6 7 8 20 101 102 103 104 2 4 6 8 10 20 ] (1x19 double)
 sThcolor: (19x3 double)
    bbox: [ -1 1 -1 1 ] (1x4 double)
 sThgeolab: []
 sThphyslab: [ 2 4 6 8 10 20 ] (1x6 double)
 sThpartlabs: []
         nq: 13258 double
   nqParents: 13258 double
 toParents: (1x1 cell)
*** nq=13258, size(z)=[13258,3]

```

eval method

Evaluates numerical datas or vectorized functions at vertices of the mesh. We denote by Th a **siMESH** object and $n_q = \text{Th.nq}$ the total number of vertices.

- `res=Th.eval(data)` : the input parameter data could be
 - a scalar,
 - a handle to a vectorized function,

- a n_q -by-1 array,
- a 1-by- m cell array of mixed previous kinds, ($m \geq 1$).

The return value is a n_q -by-1 array if the input parameter data is not a cell array otherwise it's a n_q -by- m array.

- `res=Th.eval(data,key,value,...)` specifies function options using one or more key,value pair arguments. The string key options could be
 - `d` : to specify the d-simplicial elementary meshes on which to evaluate data (default Th.d). A zero value is set on all vertices not in these elementary meshes.
 - `labels` : to specify the labels of the elementary meshes on which to evaluate data (default is all). A zero value is set on all vertices not in these elementary meshes.

Several examples are given in functions:

`fc_simesh.demos.eval2D01()`, `siMesh.demos.eval3D01()`, ...

We present now some very basic samples.

Sample 1

Listing 7: `eval` method, four ways to defined a function

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

g1=pi*ones(Th.nq,1);
g2=pi*ones(1,Th.nq);
g3=@(X) pi;

z1=Th.eval(g1);
z2=Th.eval(g2);
z3=Th.eval(g3);

fprintf(' size (z1)=[%d,%d]\n',size(z1))
fprintf(' size (z2)=[%d,%d]\n',size(z2))
fprintf(' size (z3)=[%d,%d]\n',size(z3))
fprintf(' max(abs(z2-z1))=%e\n',max(abs(z2-z1)))
fprintf(' max(abs(z3-z1))=%e\n',max(abs(z3-z1)))
```

Output

```
size(z1)=[13258,1]
size(z2)=[13258,1]
size(z3)=[13258,1]
max(abs(z2-z1))=0.000000e+00
max(abs(z3-z1))=0.000000e+00
```

Sample 2

Listing 8: `eval` method with a vector-valued function

```
meshfile=fc_oogmsh.buildmesh2d('condenser11',50,'verbose',0);
Th=siMesh(meshfile);
u=Th.feval(@(x,y) cos(3*x).*sin(4*y));
% f : R^2 -> R^3
f=@(x,y) cos(2*x).*sin(3*y),u,@(x,y) cos(4*x).*sin(5*y),pi;
z=Th.eval(f);
fprintf(' ***_nq=%d, _size(z)=[%d,%d]',Th.nq,size(z))
```

Output

```
***_nq=13258, size(z)=[13258,4]
```


get_h method

returns the maximum edges length of the mesh. We denote by Th a `siMESH` object.

- `h=Th.get_h()`

get_mesh method

Returns a vertices array q , a connectivity array me and a `toGlobal` indices array.

- `[q,me,toGlobal]=Th.get_mesh()` : returns the global vertices array q , the connectivity array me (i.e. all the $\text{Th}.d$ -simplices of the mesh). In this case, `toGlobal` is just `1:Th.nq`.
- `[q,me,toGlobal]=Th.get_mesh(key,value,...)` specifies function options using one or more key,value pair arguments. The string key options could be
 - `'d'` : to specify the d -simplicial elementary meshes to consider.
 - `'labels'` : to specify the labels of the elementary meshes to consider.

In this case, `toGlobal` is a 1-by-`length(q)` array (subset of `1:Th.nq`). If we denote by q_{glob} the global vertices array then

$$q_{\text{glob}}(:, \text{toGlobal}) == q$$

Several examples are given in functions:

`fc_simesh.demos.get_mesh2D()`, `siMesh.demos.get_mesh3D()`, `siMesh.demos.get_mesh3Ds()`

```
Listing 9: : get_mesh method, four ways to defined a function
meshfile=fc_oogmsh.buildmesh2d('condenser11',50,'verbose',0);
Th=siMesh(meshfile);

[q,me,toGlobal]=Th.get_mesh();
[q2,me2,toGlobal2]=Th.get_mesh('labels',2:2:8);
[q1,me1,toGlobal1]=Th.get_mesh('d',1,'labels',1:8);

fprintf('norm(q(:,toGlobal2)-q2,Inf)=%e\n',norm(q(:,toGlobal2)-q2,Inf))
fprintf('norm(q(:,toGlobal1)-q1,Inf)=%e\n',norm(q(:,toGlobal1)-q1,Inf))

Output
norm(q(:,toGlobal2)-q2,Inf)=0.000000e+00
norm(q(:,toGlobal1)-q1,Inf)=0.000000e+00
```

get_nme method

Returns the number of d -simplicial elements with $d = \mathcal{T}_h.d$ by default. We denote by Th a `siMESH` object.

- `nme=Th.get_nme()` : returns the number of $\text{Th}.d$ -simplicial elements in the mesh.
- `nme=Th.get_nme(key,value,...)` specifies function options using one or more key,value pair arguments. The string key options could be

- 'd' : to specify the d-simplicial elementary meshes to consider.
- 'labels' : to specify the labels of the elementary meshes to consider.

Listing 10: : get_nme method

```

meshfile=gmsht.buildmesh3d('quart_sphere2',5);
Th=siMesh(meshfile);
for d=[Th.d:-1:0]
    fprintf('Number_of_%d-simplices_: %d\n',d,Th.get_nme('d',d))
end

nme=Th.get_nme('d',2,'labels',1:4);
fprintf('Number_of_2-simplices_in_union_of_label''s_1_to_4_: %d\n',nme);

```

Output

```

[fc-oogmsh] Input file : <fc-oogmsh>/geodir/3d/quart_sphere2.geo
[fc-oogmsh] Mesh file <fc-oogmsh>/meshes/quart_sphere2-5.msh already exists.
-> Use "force" flag to rebuild if needed.
Mesh /tmp/tmp.IiKckek6YW/fc-simesh-full/fc-oogmsh-0.0.17/meshes/quart_sphere2-5.msh is a ...
3-dimensional mesh
Force dimension to 3
Number of 3-simplices : 4874
Number of 2-simplices : 1803
Number of 1-simplices : 115
Number of 0-simplices : 5
Number of 2-simplices in union of label's 1 to 4 : 788

```

get_nq method

Returns the number of vertices in the union of some elementary meshes. By default all the Th.d-simplicial elementary meshes are selected. We denote by Th a siMESH object.

- `nq=Th.get_nq()` : returns the number of vertices in the union of the Th.d-simplicial elementary meshes.
- `nq=Th.get_nq(key,value,...)` specifies function options using one or more key,value pair arguments. The string key options could be
 - 'd' : to specify the d-simplicial elementary meshes to consider.
 - 'labels' : to specify the labels of the elementary meshes to consider.

Listing 11: : get_nqe method

```

meshfile=gmsht.buildmesh3d('quart_sphere2',5);
Th=siMesh(meshfile);
for d=[Th.d:-1:0]
    fprintf('Number_of_vertices_in_%d-simplices_elementary_meshes_: ...
    %d\n',d,Th.get_nq('d',d))
end

nq=Th.get_nq('d',2,'labels',1:4);
fprintf('Number_of_vertices_in_the_union_of_2-simplices_elementary_meshes_of_...
label''s_1_to_4_: %d\n',nq);

```

Output

```

[fc-oogmsh] Input file : <fc-oogmsh>/geodir/3d/quart_sphere2.geo
[fc-oogmsh] Mesh file <fc-oogmsh>/meshes/quart_sphere2-5.msh already exists.
-> Use "force" flag to rebuild if needed.
Mesh /tmp/tmp.IiKckek6YW/fc-simesh-full/fc-oogmsh-0.0.17/meshes/quart_sphere2-5.msh is a ...
3-dimensional mesh
Force dimension to 3
Number of vertices in 3-simplices elementary meshes : 1228
Number of vertices in 2-simplices elementary meshes : 887
Number of vertices in 1-simplices elementary meshes : 111
Number of vertices in 0-simplices elementary meshes : 5
Number of vertices in the union of 2-simplices elementary meshes of label's 1 to 4 : 425

```

3.5 Hypercube as a `siMESH` object

The function `fc_simesh.HyperCube` allows to create a `siMESH` object representing an hypercube in any dimension. It uses the `FC-HYPERMESH` Matlab toolbox.

- `Th=fc_simesh.HyperCube(dim,N)` : return a `siMESH` object representing an hypercube in dimension `dim` and ...
- `Th=fc_simesh.HyperCube(dim,N,Key,Value,...)` :

2D hypercube

In Listing 12 a usage example generating a 2D hypercube as a `siMESH` object is given. This `siMESH` object is representing in Figure 4 by using the `FC-SIPLT` toolbox.

```
Listing 12 : 2D Hypercube siMESH object generated with the function siMesh.HyperCube  
Th=fc_simesh.HyperCube(2,10);  
disp(Th)
```

Output

```
siMesh with properties:  
    d: 2 double  
    dim: 2 double  
    sTh: (1x9 cell)  
    nsTh: 9 double  
    toGlobal: (1x121 double)  
    toParent: []  
    sThsimp: [ 2 1 1 1 1 0 0 0 0 ] (1x9 double)  
    sThlab: [ 1 1 2 3 4 1 2 3 4 ] (1x9 double)  
    sThcolors: (9x3 double)  
    bbox: [ 0 1 0 1 ] (1x4 double)  
    sThgeolab: []  
    sThphyslab: 1 double  
    sThpartlabs: []  
    nq: 121 double  
    nqParents: []  
    toParents: []
```

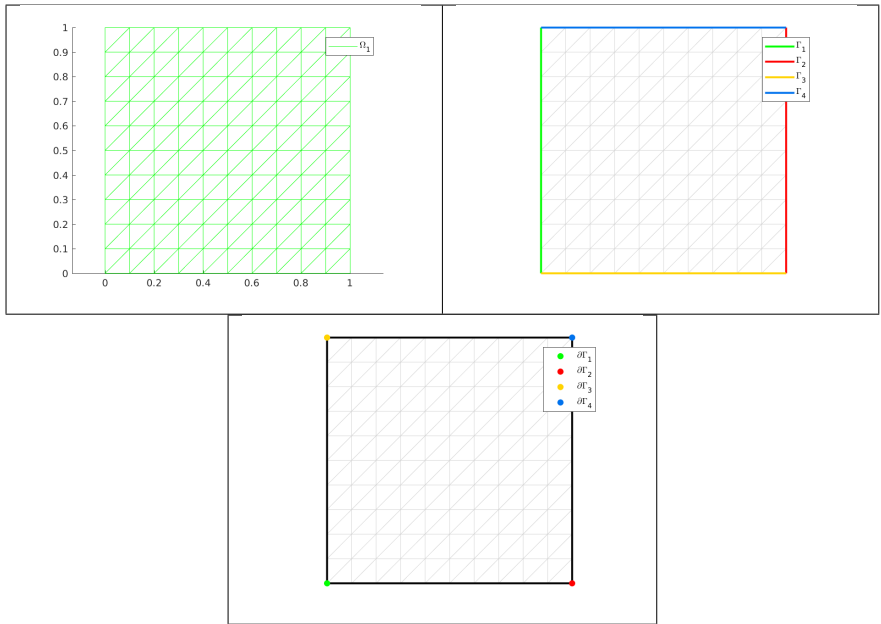


Figure 4: 2D Hypercube `siMESH` object generated with the function `fc_simesh.HyperCube`, representation of the elementary meshes with 2-simplices (top left), 1-simplices (top right) and 0-simplices (bottom)

3D hypercube

In Listing 13 a usage example generating a 3D hypercube as a `siMESH` object is given. This `siMESH` object is representing in Figure 5 by using the the `FC-SIPLT` toolbox. .

```

Listing 13: 3D Hypercube siMESH object generated with the function fc_simesh.HyperCube
Th=fc_simesh.HyperCube(3,10);
disp(Th)

Output

siMesh with properties:
    d: 3 double
    dim: 3 double
    sTh: (1x27 cell)
    nsTh: 27 double
    toGlobal: (1x1331 double)
    toParent: []
    sThsimp: (1x27 double)
    sThlab: (1x27 double)
    sThcolors: (27x3 double)
    bbox: [ 0 1 0 1 0 1 ] (1x6 double)
    sThgeolab: []
    sThphyslab: 1 double
    sThpartlabs: []
    nq: 1331 double
    nqParents: []
    toParents: []

```

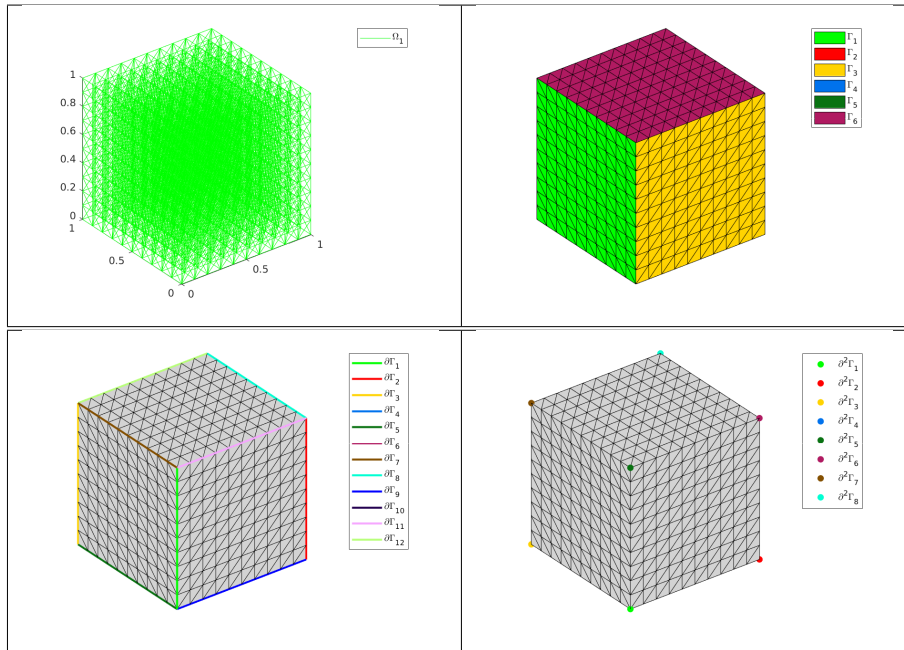


Figure 5: 3D Hypercube `siMESH` object generated with the function `siMesh.HyperCube`, representation of the elementary meshes with 3-simplices (top left), 2-simplices (top right), 1-simplices (bottom left) and 0-simplices (bottom right)

4D hypercube

In Listing 14 a usage example generating a 4D hypercube as a `siMESH` object is given.

```
Listing 14: : function fc_siMesh.HyperCube
Th=fc_siMesh.HyperCube(4,10) ;
disp(Th)

Output

siMesh with properties:
    d: 4 double
    dim: 4 double
    sTh: (1x81 cell)
    nsTh: 81 double
    toGlobal: (1x14641 double)
    toParent: []
    sThsimp: (1x81 double)
    sThlab: (1x81 double)
    sThcolors: (81x3 double)
    bbox: [ 0 1 0 1 0 1 0 1 ] (1x8 double)
    sThgeolab: []
    sThphyslab: 1 double
    sThpartlabs: []
    nq: 14641 double
    nqParents: []
    toParents: []
```

5D hypercube

In Listing 14 a usage example generating a 5D hypercube as a `siMESH` object is given.

Listing 15: : function `siMesh.HyperCube`

```
Th=fc_simesh.HyperCube(5,6);  
disp(Th)
```

Output

```
siMesh with properties:  
  d: 5 double  
  dim: 5 double  
  sTh: (1x243 cell)  
  nsTh: 243 double  
  toGlobal: (1x16807 double)  
  toParent: []  
  sThsimp: (1x243 double)  
  sThlab: (1x243 double)  
  sThcolors: (243x3 double)  
  bbox: [ 0 1 0 1 0 1 0 1 0 1 ] (1x10 double)  
  sTheolab: []  
  sThphyslab: 1 double  
  sThpartlabs: []  
  nq: 16807 double  
  nqParents: []  
  toParents: []
```