



meshtools Octave package, User's Guide*

version 0.2.0

François Cuvelier[†]

March 31, 2025

Abstract

The experimental **meshtools** Octave package contains some meshes provided by **gmsh** when exporting the built mesh as a **.m** file. These meshes can be easily used in other Octave codes for debugging or testing purpose.

0 Contents

1	Introduction	1
2	Installation	2
2.1	Installation automatic, all in one (recommanded)	2
3	Functions	3
3.1	function <code>fc_meshtools.utils.read_data_m_file</code>	3
3.2	function <code>fc_meshtools.utils.read_gmsh_m_file</code>	7
3.3	function <code>fc_meshtools.utils.get_faces</code>	8
3.4	function <code>fc_meshtools.utils.get_edges</code>	9
3.5	function <code>fc_meshtools.utils.plot_edges</code>	9
3.6	function <code>fc_meshtools.utils.get_local_mesh</code>	11
3.7	On simplicial meshes	12
3.7.1	Volumes function	12
3.7.2	Gradient of barycentric coordinates	13

1 Introduction

A mesh is given by its vertices/nodes array `q` and connectivity arrays, one by For demonstration purpose, some quadrangular and triangular/simplicial meshes are given in this package and stored in the

* $\text{\LaTeX}$ manual, revision 0.2.0, compiled with Octave 9.4.0, and packages `fc-meshtools[0.2.0]`, `fc-tools[0.1.0]`, `fc-bench[0.1.4]`, `fc-amat[0.1.4]`

[†]LAGA, UMR 7539, CNRS, Université Paris 13 - Sorbonne Paris Cité, Université Paris 8, 99 Avenue J-B Clément, F-93430 Villetaneuse, France, cuvelier@math.univ-paris13.fr.

+`fc_meshtools/data` directory. They can be load by using the function `fc_meshtools.utils.read_data_f_file`. Here are the kind of elements which can be found in the meshes present in this package:

- entity dimension $d = 0$, point,
- entity dimension $d = 1$, line,
- entity dimension $d = 2$, triangle, quadrangle,
- entity dimension $d = 3$, tetrahedron, hexahedron, prism, pyramid.

This package was only tested on Ubuntu 24.04.1 with Octave 9.3.0.

2 Installation

2.1 Installation automatic, all in one (recommanded)

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

`ofc_meshtools_install.m`

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

For example, to install this package in `~/Octave` directory, one have to copy the file `ofc_meshtools_install.m` in the `~/Octave` directory. Then in a Octave terminal run the following commands to install the `fc_meshtools` package

```
>> cd ~/Octave
>> ofc_meshtools_install
```

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

```
Parts of the <fc-meshtools> Octave package.
Copyright (C) 2018-2025 F. Cuvelier <cuvelier@math.univ-paris13.fr>

1- Downloading and extracting the packages
2- Setting the <fc-meshtools> package
Write in ~/Octave/fc-meshtools-full/fc_meshtools-0.2.0/configure_loc.m ...
3- Using packages :
->          fc-tools : 0.1.0
->          fc-bench : 0.1.4
->          fc-amat : 0.1.4
with        fc_meshtools : 0.2.0
*** Using instructions
To use the <fc-meshtools> package:
addpath('~/Octave/fc-meshtools-full/fc_meshtools-0.2.0')
fc_meshtools.init()

See ~/Octave/ofc_meshtools_set.m
```

The complete package (which included the `fc-tools` package) is stored in the directory `~/Octave/fc-meshtools-full` and, for each Octave session, one have to set the package by:

```
>> addpath('~/Octave/fc-meshtools-full/fc_meshtools-0.2.0')
>> fc_meshtools.init()
```

If it's the first time the `fc_meshtools.init()` function is used, then its output is

```
Try to use default parameters!
Use fc_tools.configure to configure.
Write in ~/Octave/fc-meshtools-full/fc_tools-0.1.0/configure_loc.m ...
Try to use default parameters!
Use fc_bench.configure to configure.
Write in ~/Octave/fc-meshtools-full/fc_bench-0.1.4/configure_loc.m ...
Try to use default parameters!
Use fc_amat.configure to configure.
Write in ~/Octave/fc-meshtools-full/fc_amat-0.1.4/configure_loc.m ...
Using fc_meshtools[0.2.0] with fc_tools[0.1.0], fc_bench[0.1.4], fc_amat[0.1.4].
```

Otherwise, the output of the `fc_meshtools.init()` function is

```
Using fc_meshtools [0.2.0] with fc_tools [0.1.0], fc_bench [0.1.4], fc_amat [0.1.4].
```

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

```
~/Octave/fc-meshtools-full
```

3 Functions

3.1 function `fc_meshtools.utils.read_data_m_file`

reads a `.m` file generated by GMSH and stored in the `data` directory of the package.

Syntaxe

```
msh=fc_meshtools.utils.read_data_m_file.read_data_m_file(Key, Value ,  
...)
```

Description

```
msh=fc_meshtools.utils.read_data_m_file.read_data_m_file(Key,Value, ...)
```

Optional key/value pairs could be used. The keys are:

- `'dim'` : to specify space dimension (2 or 3). Default 2.
- `'d'` : to specify the dimension of mesh elements (2 or 3 with `d<=dim`). Default 2.
- `'quad'` : if `true` select a quadrangular mesh otherwise a triangular mesh. Default `false`.
- `'small'` : if `true` select the small mesh otherwise a biggest mesh. Default `true`.
- `'verbose'` : if `true` display `.m` mesh file to be read. Default `true`.

Returns `msh` a structure with certain fields depending on the type of elements present in the mesh. Here are the fields always present in the `msh` structure:

- `nbNod` : number of nodes,
- `POS` : `nbNod`-by-`dim` nodes array.
- `MIN` : `[xmin,ymin]` in 2D or `[xmin,ymin,zmin]` in 3D.
- `MAX` : `[xmax,ymax]` in 2D or `[xmax,ymax,zmax]` in 3D.
- `LINES` : `nbLines`-by-3 array, `LINES(:,1:2)` is the connectivity array and `LINES(:,3)` are the line tags.

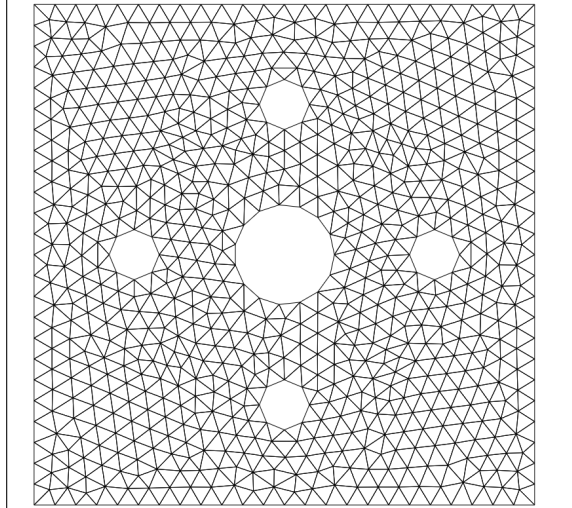
Here are the fields which can be found in structure:

- `PNT` : `nbPoints`-by-2 array (`nbPoints<=nbNod`), `PNT(i,1)` is the index in `POS` array of the point with tag `PNT(i,2)`.
- `TRIANGLES` : `nbTriangles`-by-4 array, `TRIANGLES(:,1:3)` is the connectivity array and `TRIANGLES(:,4)` are the triangle tags.
- `QUADS` : `nbQuadrangles`-by-5 array, `QUADS(:,1:4)` is the connectivity array and `QUADS(:,5)` are the quadrangle tags.
- `TETS` : `nbTetrahedra`-by-5 array, `TETS(:,1:4)` is the connectivity array and `TETS(:,5)` are the tetrahedron tags.
- `HEXAS` : `nbHexahedra`-by-9 array, `HEXAS(:,1:8)` is the connectivity array and `HEXAS(:,9)` are the hexahedron tags.
- `PYRAMIDS` : `nbPyramids`-by-6 array, `PYRAMIDS(:,1:5)` is the connectivity array and `PYRAMIDS(:,6)` are the pyramid tags.
- `PRISMS` : `nbPrisms`-by-7 array, `PRISMS(:,1:6)` is the connectivity array and `PRISMS(:,7)` are the prism tags.

Listing 1: Using `fc_meshtools.utils.read_data_m_file` function (see `fc_meshtools.demos.simplicial_2D01`). Below, the text output is given as well as the figure obtained.

```
msh=fc_meshtools.utils.read_data_m_file('d',2,'dim',2,'quad',false,'small',true);
fc_tools.utils.disp_object(msh)
patch('vertices',msh.POS,'faces',msh.TRIANGLES(:,1:3),'facecolor','None','edgecolor','k')
patch('vertices',msh.POS,'faces',msh.LINES(:,1:2),'edgecolor','b','linewidth',2)
axis equal;axis off
```

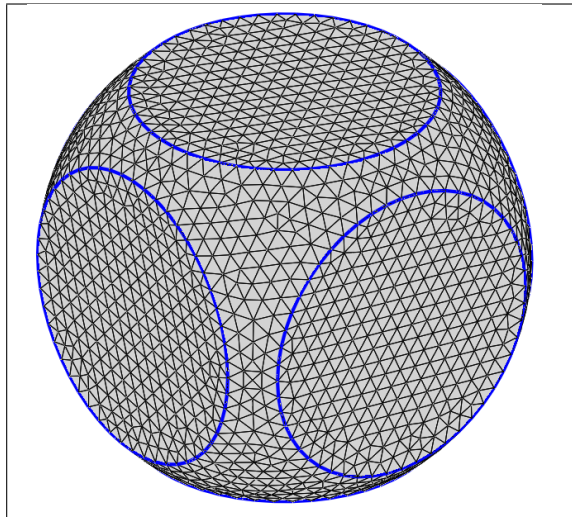
```
loading GMSH m file : <fc-meshtools>/data/condenser100_small.m
struct with fields :
  nbMod: 801 double
  POS: (801x2 double)
  MAX: [ 1 1 ] (1x2 double)
  MIN: [ -1 -1 ] (1x2 double)
  LINES: (148x3 double)
  TRIANGLES: (1467x4 double)
Save figure 1 in ./figures/simplicial_2D01_fig1_Octave940.png
[fc-tools] waiting 2(s) to finish saving figures
```



Listing 2: Using `fc_meshtools.utils.read_data_m_file` function (see `fc_meshtools.demos.simplicial_3Ds01`). Below, the text output is given as well as the figure obtained.

```
msh=fc_meshtools.utils.read_data_m_file('d',2,'dim',3,'quad',false,'small',true);
fc_tools.utils.disp_object(msh)
color=fc_tools.graphics.xcolor.val2rgb('lightgray');
patch('vertices',msh.POS,'faces',msh.TRIANGLES(:,1:end-1),'facecolor',color,'edgecolor','k')
hold on;axis equal;axis off;view(3)
%patch('vertices',msh.POS,'faces',msh.LINES(:,1:end-1),'edgecolor','b','linewidth',2)
fc_meshtools.utils.plot_edges(msh.POS,msh.LINES(:,1:2),'color','b','linewidth',2)
```

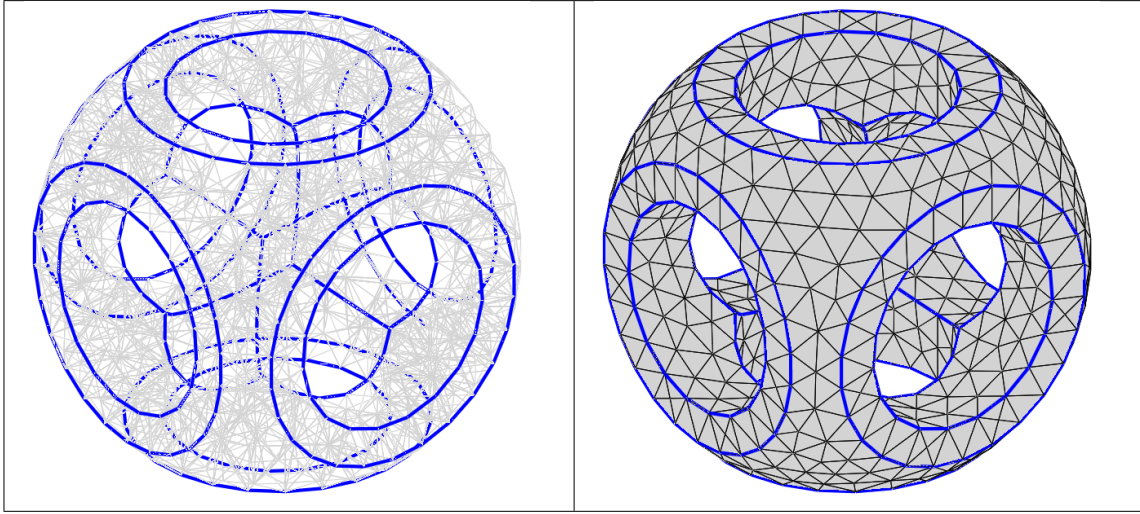
```
loading GMSH m file : <fc-meshtools>/data/dice00_small.m
struct with fields :
  nbMod: 2423 double
  POS: (2423x3 double)
  MAX: [ 0.5 0.5 0.5 ] (1x3 double)
  MIN: [ -0.5 -0.5 -0.5 ] (1x3 double)
  LINES: (312x3 double)
  TRIANGLES: (4842x4 double)
Save figure 1 in ./figures/simplicial_3Ds01_fig1_Octave940.png
[fc-tools] waiting 2(s) to finish saving figures
```



Listing 3: Using `fc_meshtools.utils.read_data_m_file` function (see `fc_meshtools.demos.simplicial_3D01`). Below, the text output is given as well as the figures obtained.

```
msh=fc_meshtools.utils.read_data_m_file('d',3,'dim',3,'quad',false,'small',true);
fc_tools.utils.disp_object(msh)
color=fc_tools.graphics.xcolor.val2rgb('lightgray');
figure(1)
patch('vertices',msh.POS,'faces',msh.TETS(:,1:end-1),'facecolor','None','edgecolor',color)
hold on
fc_meshtools.utils.plot_edges(msh.POS,msh.LINES(:,1:2),'color','b','linewidth',2)
axis equal;axis off;view(3)
figure(2)
patch('vertices',msh.POS,'faces',msh.TRIANGLES(:,1:end-1),'facecolor',color,'edgecolor','k')
hold on
fc_meshtools.utils.plot_edges(msh.POS,msh.LINES(:,1:2),'color','b','linewidth',2)
axis equal;axis off;view(3)
```

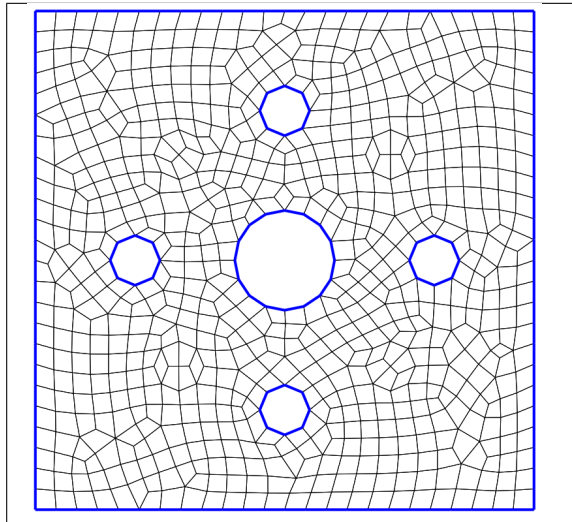
```
loading GMSH .m file : c:\fc_meshtools\data\boledice0C_small.m
struct with fields :
  nMesh: 839 double
  POS: (839x3 double)
  MAX: [ 0.5 0.5 0.5 ] (1x3 double)
  MIN: [-0.5 -0.5 -0.5 ] (1x3 double)
  LINES: (329x3 double)
  TETS: (2421x5 double)
Save figure 2 in ./figures/simplicial_3D01_fig2_Octave940.png
Save figure 1 in ./figures/simplicial_3D01_fig1_Octave940.png
[fc-tools] waiting 2(s) to finish saving figures
```



Listing 4: Using `fc_meshtools.utils.read_data_m_file` function (see `fc_meshtools.demos.quad_2D01`). Below, the text output is given as well as the figure obtained.

```
msh=fc_meshtools.utils.read_data_m_file('d',2,'dim',2,'quad',true,'small',true);
fc_tools.utils.disp_object(msh)
patch('vertices',msh.POS,'faces',msh.QUADS(:,1:end-1),'facecolor','None','edgecolor','k')
hold on
fc_meshtools.utils.plot_edges(msh.POS,msh.LINES(:,1:end-1),'color','b','linewidth',2)
axis equal;axis off
```

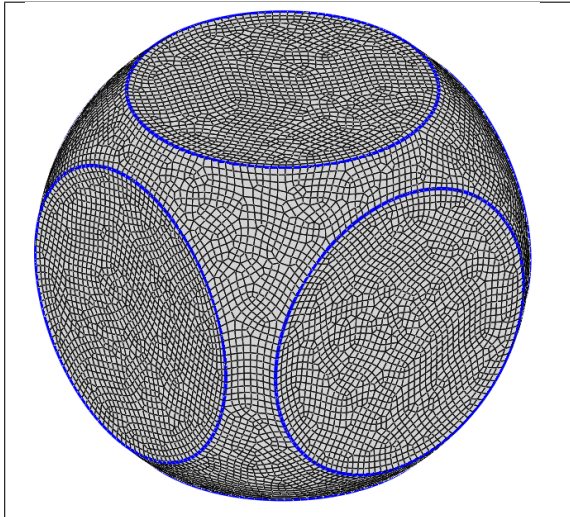
```
loading GMSH .m file : c:\fc_meshtools\data\condenser110C_quad_small.m
struct with fields :
  nMesh: 790 double
  POS: (790x2 double)
  MAX: [ 1 1 ] (1x2 double)
  MIN: [-1 -1 ] (1x2 double)
  LINES: (144x3 double)
  QUADS: (722x5 double)
Save figure 1 in ./figures/quad_2D01_fig1_Octave940.png
[fc-tools] waiting 2(s) to finish saving figures
```



Listing 5: Using `fc_meshtools.utils.read_data_m_file` function (see `fc_meshtools.demos.quad_3Ds01`). Below, the text output is given as well as the figure obtained.

```
msh=fc_meshtools.utils.read_data_m_file('d',2,'dim',3,'quad',true,'small',false);
fc_tools.utils.disp_object(msh)
color=fc_tools.graphics.xcolor.val2rgb('lightgray');
patch('vertices',msh.POS,'faces',msh.QUADS(:,1:4),'facecolor',color,'edgecolor','k')
hold on
fc_meshtools.utils.plot_edges(msh.POS,msh.LINES(:,1:2),'color','b','linewidth',2)
axis equal;axis off;view(3)
```

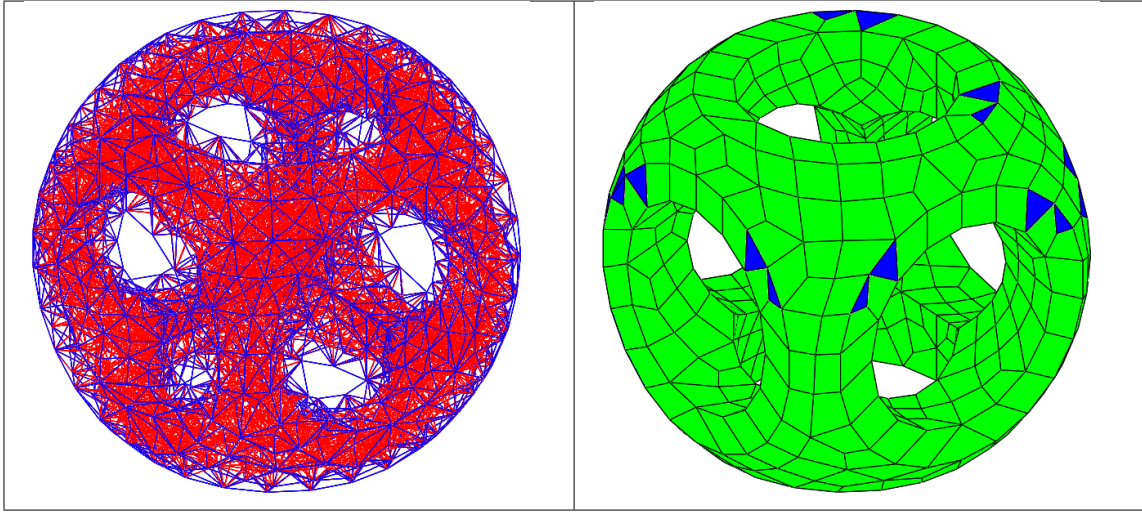
```
loading QMSH .m file : c:\fc-meshtools\data\dice0C_quad.m
struct with fields :
  mshMod: 14076 double
  POS: (14076x3 double)
  MAX: [ 0.5 0.5 0.5 ] (1x3 double)
  MIN: [-0.5 -0.5 -0.5 ] (1x3 double)
  LINES: (768x3 double)
  QUADS: (14074x5 double)
Save figure 1 in :./figures/quad_3Ds01_fig1_Octave940.png
[fc-tools] waiting 2(s) to finish saving figures
```



Listing 6: Using `fc_meshtools.utils.read_data_m_file` function (see `fc_meshtools.demos.quad_3D01`). Below, the text output is given as well as the figures obtained.

```
msh=fc_meshtools.utils.read_data_m_file('d',3,'dim',3,'quad',true,'small',true);
fc_tools.utils.disp_object(msh)
colors=fc_tools.graphics.selectColors(3);
figure(1)
FacesTETS=fc_meshtools.utils.get_faces(msh.TETS(:,1:end-1),'tetrahedron');
patch('vertices',msh.POS,'faces',FacesTETS,'facecolor','None','edgecolor','r')
FacesPYR=fc_meshtools.utils.get_faces(msh.PYRAMIDS(:,1:end-1),'pyramid');
patch('vertices',msh.POS,'faces',FacesPYR,'facecolor','None','edgecolor','b')
axis equal;axis off;view(3)
figure(2)
patch('vertices',msh.POS,'faces',msh.TRIANGLES(:,1:end-1),'facecolor',colors(1,:), 'edgecolor','k')
patch('vertices',msh.POS,'faces',msh.QUADS(:,1:end-1),'facecolor',colors(2,:), 'edgecolor','k')
axis equal;axis off;view(3)
```

```
loading GMSH .m file : c:\fc_meshtools\data\boleddice0C_quad_small.m
struct with fields :
  nbNod: 3129 double
  POS: (3129x3 double)
  MAX: [ 0.5 0.5 0.5 ] (1x3 double)
  MIN: [-0.5 -0.5 -0.5 ] (1x3 double)
  LINES: (306x3 double)
  TRIANGLES: (30x4 double)
  QUADS: (722x5 double)
  TETS: (1665x5 double)
  PYRAMIDS: (722x5 double)
Save figure 2 in ./figures/quad_3D01_fig2_Octave940.png
Save figure 1 in ./figures/quad_3D01_fig1_Octave940.png
[fc-tools] waiting 2(s) to finish saving figures
```



3.2 function `fc_meshtools.utils.read_gmsh_m_file`

reads a `.m` file generated by GMSH.

Syntaxe

```
msh=fc_meshtools.utils.read_data_m_file.read_gmsh_m_file(filename)
```

Description

```
msh=fc_meshtools.utils.read_gmsh_m_file.read_gmsh_m_file(filename)
```

Returns `msh` a structure with certain fields depending on the type of elements present in the mesh. Here are the fields always present in the `msh` structure:

- `nbNod` : number of nodes,
- `POS` : `nbNod`-by-`dim` nodes array.
- `MIN` : `[xmin,ymin]` in 2D or `[xmin,ymin,zmin]` in 3D.
- `MAX` : `[xmax,ymax]` in 2D or `[xmax,ymax,zmax]` in 3D.
- `LINES` : `nbLines`-by-3 array, `LINES(:,1:2)` is the connectivity array and `LINES(:,3)` are the line tags.

Here are the principal fields which can be found in the structure:

- `PNT` : `nbPoints`-by-2 array (`nbPoints` ≤ `nbNod`), `PNT(i,1)` is the index in `POS` array of the point with tag `PNT(i,2)`.

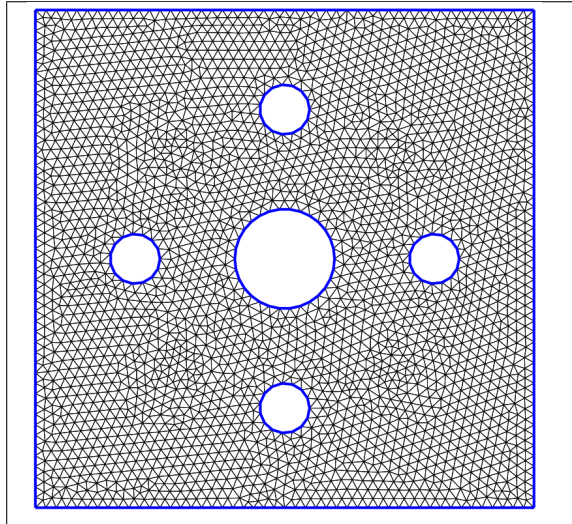
- **TRIANGLES** : nbTriangles-by-4 array, TRIANGLES(:,1:3) is the connectivity array and TRIANGLES(:,4) are the triangle tags.
- **QUADS** : nbQuadrangles-by-5 array, QUADS(:,1:4) is the connectivity array and QUADS(:,5) are the quadrangle tags.
- **TETS** : nbTetrahedra-by-5 array, TETS(:,1:4) is the connectivity array and TETS(:,5) are the tetrahedron tags.
- **HEXAS** : nbHexahedra-by-9 array, HEXAS(:,1:8) is the connectivity array and HEXAS(:,9) are the hexahedron tags.
- **PYRAMIDS** : nbPyramids-by-6 array, PYRAMIDS(:,1:5) is the connectivity array and PYRAMIDS(:,6) are the pyramid tags.
- **PRISMS** : nbPrisms-by-7 array, PRISMS(:,1:6) is the connectivity array and PRISMS(:,7) are the prism tags.

With an n th order mesh, other fields are present in the .m file: see the source file of GMSH `src/geo/GModelIO_MATLAB.cpp`

Listing 7: Using `fc_meshtools.utils.read_gmsh_m_file` function (see `fc_meshtools.demos.read_gmsh_m_file_2D01`). Below, the text output is given as well as the figure obtained.

```
env=fc_meshtools.environment();
filename=fullfile(env.data_dir,'condenser11OC.m');
msh=fc_meshtools.utils.read_gmsh_m_file(filename);
fc_tools.utils.disp_object(msh)
patch('vertices',msh.POS,'faces',msh.TRIANGLES(:,1:3),'facecolor','None','edgecolor','k')
hold on
fc_meshtools.utils.plot_edges(msh.POS,msh.LINES(:,1:end-1),'color','b','linewidth',2)
axis equal;axis off
```

```
struct with fields :
  nbNodes: 2771 double
  POS: (2771x2 double)
  MAX: [ 1 1 ] (1x2 double)
  MIN: [-1 -1 ] (1x2 double)
  LINES: (282x3 double)
  TRIANGLES: (6268x4 double)
Save figure 1 in ./figures/read_gmsh_m_file_2D01_fig1_Octave940.png
[fc-tools] waiting 2(s) to finish saving figures
```



3.3

function `fc_meshtools.utils.get_faces`

Returns the faces of all mesh elements of type `geo` and given by its connectivity array `connect`.

Syntaxe

```
Faces=fc_meshtools.utils.get_faces(connect,geo)
Faces=fc_meshtools.utils.get_faces(connect,geo,'unique',true)
```

Description

```
Faces=fc_meshtools.utils.get_faces(connect,geo)
```


- `connect` : is an `nme-by-nv` array with `nme`, the number of mesh elements, and `nv` the number of vertices of an element of type `geo`
- `geo` : type of mesh elements. Must be in

`{'triangle ','quadrangle','tetrahedron ','hexahedron','prism ','pyramid'}`

The index order of the vertices of an element are those of `gmsh`.

The output `Faces` is an `nf-by-m` connectivity array where `m` is the maximum number of vertices by faces `m:3` for triangle,tetrahedron, `m:4` for quadrangle,hexahedron,prism,pyramid For prisms and pyramids, there are both triangular and quadrangular faces. The 3 indices for triangular faces with `m==4`, are in `Faces(i,1:3)` and we have `Faces(i,4)==NaN`.

An optional key/value pair could be used. The key is:

- `'unique'`: logical, if `true` the returned `Faces` are unique (always unique with line,triangle, quadrangle). Default `false`.

3.4 function `fc_meshtools.utils.get_edges`

Returns the edges of all mesh elements of type `geo` and given by its connectivity array `connect`.

Syntaxe

```
Edges=fc_meshtools.utils.get_edges(connect,geo)
Edges=fc_meshtools.utils.get_edges(connect,geo,'unique',true)
```

Description

```
Edges=fc_meshtools.utils.get_edges(connect,geo)
```

- `connect` : is an `nme-by-nv` array with `nme`, the number of mesh elements, and `nv` the number of vertices of an element of type `geo`
- `geo` : type of mesh elements. Must be in

`{'triangle ','quadrangle','tetrahedron ','hexahedron','prism ','pyramid'}`

The index order of the vertices of an element are those of `gmsh`.

The output `Edges` is an `ned-by-2` connectivity array where `ned` is the number of edges.

An optional key/value pair could be used. The key is:

- `'unique'`: logical, if `true` the returned `Edges` are unique. Default `false`.

3.5 function `fc_meshtools.utils.plot_edges`

Plots edges given by a vertex array `vertices` and a connectivity array `connect`

Syntaxe

```
fc_meshtools.utils.get_edges(vertices,connect, key,value, ...)
h=fc_meshtools.utils.get_edges(vertices,connect, key,value, ...)
```

Description

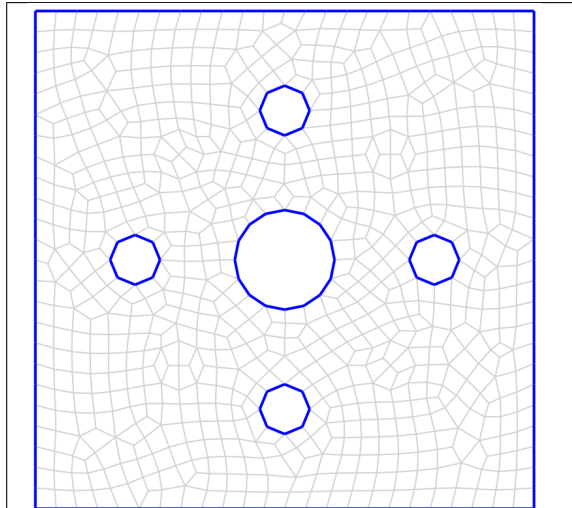
```
fc_meshtools.utils.get_edges(vertices,connect, key,value, ...)
```

- **vertices** : **nv-by-dim** array, **nv** number of vertices, **dim** space dimension (2 or 3).
- **connect** : **N-by-2** array with **N**, the number of edges, such that the two vertices of the **i**-th edges are **vertices(connect(i,1),:)** and **vertices(connect(i,2),:)** .

Optional **key/value** pairs are transfered to the **plot3** function in dimension 3 or to the **plot** function in dimension 2.

If requested, the output value is that of **plot** or **plot3** function.

```
Listing 8: Using fc_meshtools.utils.get_edges and fc_meshtools.utils.plot_edges and functions (see fc_meshtools.demos.quad_2D02).
msh=fc_meshtools.utils.read_data_m_file('d',2,'dim',2,'quad',true,'small',true);
fc_tools.utils.disp_object(msh)
Edges=fc_meshtools.utils.get_edges(msh.QUADS(:,1:end-1),'quadrangle','unique',true);
fc_meshtools.utils.plot_edges(msh.POS,Edges,'color','lightgray','linewidth',1)
hold on
fc_meshtools.utils.plot_edges(msh.POS,msh.LINES(:,1:end-1),'color','b','linewidth',2)
axis equal;axis off
```

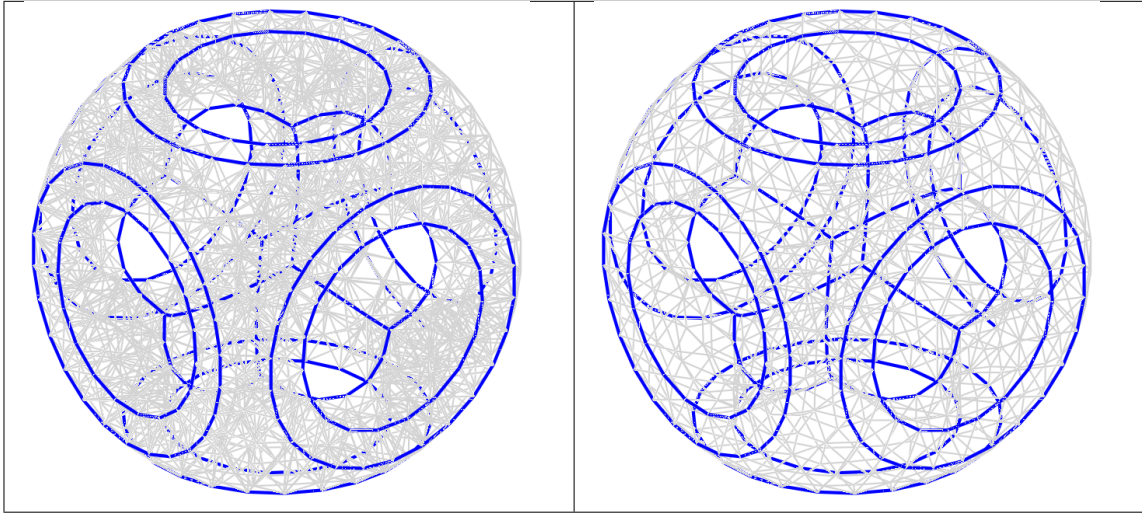


Listing 9: Using `fc_meshtools.utils.get_edges` and `fc_meshtools.utils.plot_edges` and functions (see `fc_meshtools.demos.simplicial_3D02`).

```
msh=fc_meshtools.utils.read_data_m_file('d',3,'dim',3,'quad',false,'small',true);
fc_meshtools.utils.disp_object(msh)

EdgesTETS=fc_meshtools.utils.get_edges(msh.TETS(:,1:end-1),'tetrahedron','unique',true);
figure(1)
fc_meshtools.utils.plot_edges(msh.POS,EdgesTETS,'color','lightgray','linewidth',1)
hold on;axis equal;axis off;view(3)
fc_meshtools.utils.plot_edges(msh.POS,msh.LINES(:,1:end-1),'color','b','linewidth',2)

EdgesTRI=fc_meshtools.utils.get_edges(msh.TRIANGLES(:,1:end-1),'triangle','unique',true);
figure(2)
fc_meshtools.utils.plot_edges(msh.POS,EdgesTRI,'color','lightgray','linewidth',1)
hold on;axis equal;axis off;view(3)
fc_meshtools.utils.plot_edges(msh.POS,msh.LINES(:,1:end-1),'color','b','linewidth',2)
```



3.6

function `fc_meshtools.utils.get_local_mesh`

Syntax Returns a local mesh associated with `msh`, a structure provided by a `.m` file generated with `gmsh`.

Description

```
function [q,me,L2G]=get_local_mesh(msh,type)
```

- `msh` : the structure obtain when loading a `.m` file generated by `gmsh`.
- `type` : one of the `type` of elements present in the `msh` structure as field. Could be `'LINES'`, `'TRIANGLES'`, `'QUADS'`, `'TETS'`, ...

An optional key/value pair can be used. The key is

- `'tags'` : to select `type` of elements with tag in provided value.

Returns three arrays:

- `'q'` : nodes array with dimension `dim-by-nq`,
- `'me'` : connectivity array of dimension `m-by-nme` (`m`: number of vertices by element, `nme`: number of `type` elements) associated with the array `q`. The vertices of the `k`-th `type` element are

$$q(:,me(k,1)), \dots, q(:,me(k,m))$$

- `'L2G'` : local to global index.

$$q(:,i) == msh.POS(L2G(i),:)$$

One can refer to `fc_meshtools.demos.get_local_mesh_*` programs to see multiple usages. An example is given in Listing 10.

Listing 10: Using `fc_meshtools.utils.get_local_mesh` function (see `fc_meshtools.demos.get_local_mesh_2D01`). Below, the text output is given.

```
msh=fc_meshtools.utils.read_data_m_file('d',2,'dim',2,'quad',false,'small',true);
Stags=fc_meshtools.utils.get_tags(msh);

[q,me,L2G]=fc_meshtools.utils.get_local_mesh(msh,'TRIANGLES');
fprintf('TRIANGLES elements [nme=%d] :\n',size(me,2))
for k=1:size(me,1)
    E=norm(abs(q(:,me(k,:)))-msh.POS(L2G(me(k,:)),:),Inf);
    fprintf(' Compare the vertex %d of all elements, error=%5e\n',k,E)
end

n=numel(Stags.LINES);
for i=1:n
    fprintf('LINES elements with tag=%d [nme=%d] :\n',Stags.LINES(i),size(me,2))
    [q,me,L2G]=fc_meshtools.utils.get_local_mesh(msh,'LINES','tags',Stags.LINES(i));
    for k=1:size(me,1)
        E=norm(abs(q(:,me(k,:)))-msh.POS(L2G(me(k,:)),:),Inf);
        fprintf(' Compare the vertex %d of all elements, error=%5e\n',k,E)
    end
end

loading GMSH .m file : <fc-meshtools>/data/condenser110C_small.m
TRIANGLES elements [nme=1467] :
 Compare the vertex 1 of all elements, error=0.00000e+00
 Compare the vertex 2 of all elements, error=0.00000e+00
 Compare the vertex 3 of all elements, error=0.00000e+00
LINES elements with tag=1 [nme=1467] :
 Compare the vertex 1 of all elements, error=0.00000e+00
 Compare the vertex 2 of all elements, error=0.00000e+00
LINES elements with tag=10 [nme=96] :
 Compare the vertex 1 of all elements, error=0.00000e+00
 Compare the vertex 2 of all elements, error=0.00000e+00
LINES elements with tag=21 [nme=16] :
 Compare the vertex 1 of all elements, error=0.00000e+00
 Compare the vertex 2 of all elements, error=0.00000e+00
LINES elements with tag=22 [nme=8] :
 Compare the vertex 1 of all elements, error=0.00000e+00
 Compare the vertex 2 of all elements, error=0.00000e+00
LINES elements with tag=23 [nme=8] :
 Compare the vertex 1 of all elements, error=0.00000e+00
 Compare the vertex 2 of all elements, error=0.00000e+00
LINES elements with tag=24 [nme=8] :
 Compare the vertex 1 of all elements, error=0.00000e+00
 Compare the vertex 2 of all elements, error=0.00000e+00
[fc-tools] waiting 2(s) to finish saving figures
```

3.7 On simplicial meshes

The followings functions can be used on simplicial meshes (i.e. meshes composed with triangular elements or with tetrahedron elements)

3.7.1 Volumes function

Syntaxe Returns all the element volumes of a mesh given by a `dim`-by-`nq` vertices array `q` and a connectivity array `me` of dimension `(dim+1)`-by-`nme`. One can refer to [1] for computational details.

Description

```
vols=fc_meshtools.simplicial.Volumes(q,me)
```

`vols(k)` is the volume of the `k`-th mesh element where its vertices are the columns of `q(:,me(:,k))`.

In Listing 11, some examples are provided.

Listing 11: : examples of `fc_meshtools.simplicial.Volumes` function usage

```
msh2D=fc_meshtools.utils.read_data_m_file('d',2,'dim',2,'box',true);
q2=msh2D.POS';me2=msh2D.TRIANGLES(:,1:3)';
vols2=fc_meshtools.simplicial.Volumes(q2,me2);
fprintf('area=%16e\n',sum(vols2))
msh3D=fc_meshtools.utils.read_data_m_file('d',3,'dim',3,'box',true);
q3=msh3D.POS';me3=msh3D.TETS(:,1:4)';
vols3=fc_meshtools.simplicial.Volumes(q3,me3);
fprintf('volume=%16e\n',sum(vols3))
msh3Ds=fc_meshtools.utils.read_data_m_file('d',2,'dim',3,'box',true);
q3Ds=msh3Ds.POS';me3Ds=msh3Ds.TRIANGLES(:,1:3)';
vols3Ds=fc_meshtools.simplicial.Volumes(q3Ds,me3Ds);
fprintf('surface=%16e\n',sum(vols3Ds))
```

Output

```
loading GMSH .m file : <fc-meshtools>/data/box2D_small.m
area=1.000000000000000e+00
loading GMSH .m file : <fc-meshtools>/data/box3D_small.m
volume=9.999999999999999e-01
loading GMSH .m file : <fc-meshtools>/data/box3Ds_small.m
surface=5.999999999999998e+00
```

3.7.2 Gradient of barycentric coordinates

Syntaxe Returns all the gradients of barycentric coordinates of each element of a mesh given by a dim -by- n_q vertices array q and a connectivity array me of dimension $(\text{dim}+1)$ -by- n_{me} . One can refer to [1] for computationnal details.

Description

$G = \text{fc_meshtools.simplicial.GradBaCo}(q, me)$

$G(k, i, :)$ is the gradient of the i -th barycentric coordinate of the k -th mesh element.

In Listing 12, some examples are provided.

Listing 12: : examples of `fc_meshtools.simplicial.GradBaCo` function usage

```
msh2D=fc_meshtools.utils.read_data_m_file('d',2,'dim',2,'box',true);
q2=msh2D.POS';me2=msh2D.TRIANGLES(:,1:3)';
G2=fc_meshtools.simplicial.GradBaCo(q2,me2);
whos('q2','me2','G2')
```

Output

```
loading GMSH .m file : <fc-meshtools>/data/box2D_small.m
Variables visible from the current scope:

variables in scope: top scope

Attr  Name      Size      Bytes Class
====  =====  =====  =====
      q2        2x107      1712 double
      me2       3x180      4320 double
      G2      180x3x2      8640 double

Total is 1834 elements using 14672 bytes
```

3 References

- [1] F. Cuvelier. Exact integration for products of power of barycentric coordinates over d -simplexes in R^n . <http://hal.archives-ouvertes.fr/hal-00931066v1>, June 2018. preprint.

Informations for git maintainers of the Octave package

git informations on the packages used to build this manual

```
-----
name : fc-meshtools
tag : 0.2.0
commit : 3e2d4d43f99de8f8c5f4b69580f4230261bb7dea
date : 2025-03-31
time : 11-13-35
status : 0
-----
name : fc-tools
tag : 0.1.0
commit : 8781a7e64952b4f2c3684d97ad4e78ec1cbe2fdc
date : 2025-03-31
time : 11-14-20
status : 0
-----
name : fc-bench
tag : 0.1.4
commit : 25e94e40535f90958a7ae9ddcb8fef557487bd9b
date : 2025-03-28
time : 12-47-56
status : 0
-----
name : fc-amat
tag : 0.1.4
commit : 63a829130f4412773dbb4a35b6461daa7fc5d40a
date : 2025-03-28
time : 12-55-26
status : 0
-----
```

git informations on the L^AT_EX package used to build this manual

```
-----
name : fctools
tag :
commit : 03d38737a795cdbf4e1a8754470e963cdf83316
date : 2025-01-24
time : 09:58:52
status : 1
-----
```

Using the remote configuration repository:

```
url      ssh://lagagit/MCS/Cuvelier/Matlab/fc-config
commit   43694ef530ee6ca83f9472051ee507ce19bda38d
```