




 **meshtools** Octave package, User's Guide*
version 0.1.0

François Cuvelier[†]

March 22, 2019

Abstract

The experimental  Octave package contains some simplicial meshes given by their vertices array **q** and connectivity array **me**. These meshes can be easily used in other Octave codes for debugging or testing purpose.

0 Contents

1	Introduction	2
2	Installation	2
2.1	Installation automatic, all in one (recommanded)	2
3	Simplicial meshes	3
4	Functions	4
4.1	getMesh functions	4
4.2	Volumes function	5
4.3	Gradient of barycentric coordinates	5

* \LaTeX manual, revision 0.1.0, compiled with Octave 5.1.0, and packages `fc-meshtools[0.1.0]`, `fc-tools[0.0.27]`

[†]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.

1 Introduction

A simplicial mesh is given by its vertices array **q** and its connectivity array **me**. For demonstration purpose, some simplicial meshes are given in this package and stored in the `+fc_meshtools/data` directory. They can be load by using the functions `fc_meshtools.simplicial.getMesh2D`, `fc_meshtools.simplicial.getMesh3D` or `fc_meshtools.simplicial.getMesh3Ds`. Here are the kind of simplicial meshes present in this package:

- a triangular mesh in dimension 2, made with 2-simplices (ie. triangles),
- a tetrahedral mesh in dimension 3, made with 3-simplices (ie. tetrahedron),
- a triangular mesh in dimension 3 (surface mesh), made with 2-simplices,
- a line mesh in dimension 2 or 3 made with 1-simplices (ie. lines).

This package was tested on various OS with Octave releases:

Operating system	4.2.0	4.2.1	4.2.2	4.4.0	4.4.1	5.1.0
CentOS 7.6.1810	✓	✓	✓	✓	✓	✓
Debian 9.8	✓	✓	✓	✓	✓	✓
Fedora 29	✓	✓	✓	✓	✓	✓
OpenSUSE Leap 15.0	✓	✓	✓	✓	✓	✓
Ubuntu 18.04.2 LTS	✓	✓	✓	✓	✓	✓
MacOS High Sierra 10.13.6			✓	✓	✓	
MacOS Mojave 10.14				✓	✓	
Windows 10 (1803)	✓	✓	✓	✓	✓	✓

It is not compatible with Octave releases prior to 4.2.0.

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-mesh* and the required *fc-tools* package in the current directory.

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

```
>> cd ~/Octave/packages
>> 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) 2018-2019 F. Cuvelier <cuvelier@math.univ-paris13.fr>

1- Downloading and extracting the packages
2- Setting the <fc-meshtools> package
Write in ~/Octave/packages/fc-meshtools-full/fc_meshtools-0.1.0/
  configure_loc.m ...
3- Using packages :
  ->          fc-tools : 0.0.26
*** Using instructions
To use the <fc-meshtools> package:
  addpath('~/Octave/packages/fc-meshtools-full/fc_meshtools-0.1.0')
  fc_meshtools.init()

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

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

```
>> addpath('~/Octave/packages/fc-meshtools-full/fc_meshtools-0.1.0')
>> 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/packages/fc-meshtools-full/fc_tools-0.0.26/
  configure_loc.m ...
Using fc_meshtools[0.1.0] with fc_tools[0.0.26].
```

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

```
Using fc_meshtools[0.1.0] with fc_tools[0.0.26].
```

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

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

3 Simplicial meshes

The functions `fc_meshtools.simplicial.getMesh2D`, `gfc_meshtools.simplicial.etMesh3D` and `fc_meshtools.simplicial.getMesh3Ds` return a mesh vertices array \mathbf{q} , a mesh elements connectivity array associated with the input argument d (simplex dimension) and the indices array `toGlobal`. The vertices array \mathbf{q} is a dim -by- n_q array where dim is the space dimension (2 or 3) and n_q the number of vertices. The connectivity array \mathbf{me} is a $(d + 1)$ -by- n_{me} array where n_{me} is the number of mesh elements and $0 \leq d \leq dim$ is the simplicial dimension:

- $d = 0$: points,
- $d = 1$: lines,
- $d = 2$: triangle,
- $d = 3$: tetrahedron.

So we can use these functions to obtain

- 3D mesh: `getMesh3D(3)` (*main* mesh), `getMesh3D(2)`, `getMesh3D(1)`, `getMesh3D(0)`,
- 3D surface mesh: `getMesh3Ds(2)` (*main* mesh), `getMesh3Ds(1)`, `getMesh3Ds(0)`,
- 2D mesh: `getMesh2D(2)` (*main* mesh), `getMesh2D(1)`, `getMesh2D(0)`.

For example,

- `[q3,me3,toGlobal3]=fc_meshtools.simplicial.getMesh3D(3)` return a 3-simplicial mesh (*main* mesh) in space dimension $dim = 3$,
- `[q2,me2,toGlobal2]=fc_meshtools.simplicial.getMesh3D(2)` return a 2-simplicial mesh in space dimension $dim = 3$.

The third output are indices of the vertices in the *main* mesh:
`q3(:,toGlobal2) == q2`

4 Functions

4.1 getMesh functions

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

Description

```
[q,me,toGlobal]=fc_meshtools.simplicial.getMesh3D(d)
```

```
[q,me,toGlobal]=fc_meshtools.simplicial.getMesh3Ds(d)
```

```
[q,me,toGlobal]=fc_meshtools.simplicial.getMesh2D(d)
```

Returns a vertices array `q`, a connectivity array `me` and an indices array `toGlobal` depending on the value of the `d`. For a 3D mesh, $d \in \llbracket 0, 3 \rrbracket$, and for a 2D or 3Ds mesh, $d \in \llbracket 0, 2 \rrbracket$.

In Listing 2, some examples are provided.

Listing 1: : examples of `fc_meshtools.simplicial.getMesh3D` function usage

```
[q2,me2,toG2]=fc_meshtools.simplicial.getMesh3D(2);
[q3,me3,toG3]=fc_meshtools.simplicial.getMesh3D(3);
whos('q2','me2','toG2','q3','me3','toG3')
fprintf('Error: %5e\n',norm(q3(:,toG2) - q2,Inf))
```

Output

```
Variables in the current scope:

Attr Name      Size           Bytes Class
=====
q2              3x7533         180792 double
me2             3x15074        361776 double
toG2            1x7533         60264 double
q3              3x17416        417984 double
me3             4x84302        2697664 double
toG3            1x17416        139328 double

Total is 482226 elements using 3857808 bytes

Error: 0.00000e+00
```

4.2 Volumes function

Syntaxe Returns all the element volumes of a mesh given by a vertices array `q` and a connectivity array `me`. 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 2, some examples are provided.

Listing 2: : examples of `fc_meshtools.simplicial.Volumes` function usage

```
[q,me,toG]=fc_meshtools.simplicial.getMesh3D(2);
vols=fc_meshtools.simplicial.Volumes(q,me);
whos('q','me','toG','vols')
```

Output

```
Variables in the current scope:

Attr Name      Size           Bytes Class
=====
q               3x7533         180792 double
me              3x15074        361776 double
toG             1x7533         60264 double
vols            1x15074        120592 double

Total is 90428 elements using 723424 bytes
```

4.3 Gradient of barycentric coordinates

Syntaxe Returns all the gradients of barycentric coordinates of each element of a mesh given by a vertices array `q` and a connectivity array `me`. One can refer to [1] for computational details.

Description

```
G=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 3, some examples are provided.

Listing 3: : examples of `fc_meshtools.simplicial.GradBaCo` function usage

```
[q,me,toG]=fc_meshtools.simplicial.getMesh3D(3);  
G=fc_meshtools.simplicial.GradBaCo(q,me);  
whos('q','me','toG','G')
```

Output

Variables in the current scope:

Attr Name	Size	Bytes	Class
====	====	====	====
q	3x17416	417984	double
me	4x84302	2697664	double
toG	1x17416	139328	double
G	84302x3x4	8092992	double

Total is 1418496 elements using 11347968 bytes

4 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.