oogmsh Matlab toolbox, User's Guide*

version 0.3.1

François Cuvelier†

January 28, 2025

Abstract

This Matlab toolbox make it possible to generate mesh files from *.geo* files by using `gmsh`. It's also possible with the ooGmsh2 and ooGmsh4 classes to read the mesh file (respectively for MSH file format version `2.2` and version `4.x`).This toolbox must be regarded as a very simple interface between gmsh files and Matlab. So you are free to create any data structures or objects you want from an ooGmsh2 object or an ooGmsh4 object.

0 Contents

*LaTeX manual, revision 0.3.1, compiled with Matlab 2022a, and toolboxes `fc-oogmsh`[0.3.1], `fc-tools`[0.0.36], `fc-bench`[0.1.4], `fc-amat`[0.1.4], `fc-meshtools`[0.1.5], `fc-graphics4mesh`[0.1.7], and using `gmsh` 4.13.1

†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.

This work was supported by the ANR project DEDALES under grant ANR-14-CE23-0005.

1 Introduction

The oogmsh Matlab toolbox is closely related to **gmsh**, see [2] or [3], which is a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. **gmsh** can also build two-dimensional meshes and three-dimensional surface meshes. This toolbox was initialy created to make it possible from Matlab to rapidly

- generate mesh file from .geo file by using **gmsh**
- efficiently read this mesh file and store its contents in ooGmsh Matlab object easy to manipulate.

The ooGmsh Matlab object can be used to create, from a .msh file, any data structures or objects needed by your project. For example, the fc-simesh Matlab toolbox uses this toolbox to create the siMesh object containing all the simplices elements of the mesh.

This toolbox was only tested on Ubuntu 24.04.1 with Matlab R2022a and **gmsh** 4.13.1

Firstly, we explain how to configure the oogmsh toolbox for using **gmsh**. Thereafter, we describe the oogmsh 's functions which use **gmsh** to create mesh files.

2 Installation

2.1 Installation automatic, all in one (recommanded)

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

mfc_oogmsh_install.m

or get it on the dedicated web page. Thereafter, one run it under Matlab. This command download, extract and configure the *fc-oogmsh* toolbox and the required toolboxes *fc-tools*, *fc-meshtools* and *fc-graphics4mesh*, in the current directory.

By default, the gmsh binary is supposed to be located in

- `<USERDIR>/bin/gmsh` under linux,
- `<USERDIR>/GMSH/Gmsh.app/Contents/MacOS/gmsh` under Mac OS X,
- `<USERDIR>/Softwares/GMSH/gmsh.exe` under Windows

To specify an other location one can do

```
>> mfc_oogmsh_install('gmsh_bin', GMSH)
```

where `GMSH` is the gmsh binary with path as a string. It's also possible, after installation, to change the gmsh binary by using the Matlab command

```
>> fc_oogmsh.configure('gmsh_bin','~/gmsh-4.13.1/bin/gmsh')
```

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

```
>> cd ~/Matlab
>> mfc_oogmsh_install
```

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

```
Parts of the <fc-oogmsh> Matlab toolbox.
Copyright (C) 2017-2025 F. Cuvelier

1- Downloading and extracting the toolboxes
2- Setting the <fc-oogmsh> toolbox
Write in ~/Matlab/fc-oogmsh-full/fc_oogmsh-0.3.1/configure_loc.m ...
3- Using toolboxes :
   ->             fc-tools : 0.0.36
   ->             fc-bench : 0.1.4
   ->              fc-amat : 0.1.4
   ->         fc-meshtools : 0.1.5
   ->     fc-graphics4mesh : 0.1.7
 with            fc-oogmsh : 0.3.1
*** Using instructions
   To use the <fc-oogmsh> toolbox:
   addpath('~/Matlab/fc-oogmsh-full/fc_oogmsh-0.3.1')
   fc_oogmsh.init()

   See ~/Matlab/mfc_oogmsh_set.m
```

The complete toolbox (i.e. with all the other needed toolboxes) is stored in the directory

`~/Matlab/fc-oogmsh-full`

and, for each Matlab session, one has to set the toolbox by:

```
>> addpath('~/Matlab/fc-oogmsh-full/mfc-oogmsh-0.3.1')
>> fc_oogmsh.init()
```

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

```
Try to use default parameters!
 Use fc_tools.configure to configure.
Write in ~/Matlab/fc-oogmsh-full/fc_tools-0.0.36/configure_loc.m ...
Try to use default parameters!
 Use fc_bench.configure to configure.
Write in ~/Matlab/fc-oogmsh-full/fc_bench-0.1.4/configure_loc.m ...
Try to use default parameters!
 Use fc_amat.configure to configure.
Write in ~/Matlab/fc-oogmsh-full/fc_amat-0.1.4/configure_loc.m ...
Try to use default parameters!
 Use fc_meshtools.configure to configure.
Write in ~/Matlab/fc-oogmsh-full/fc_meshtools-0.1.5/configure_loc.m ...
Try to use default parameters!
 Use fc_graphics4mesh.configure to configure.
Write in ~/Matlab/fc-oogmsh-full/fc_graphics4mesh-0.1.7/configure_loc.m ...
Using fc_oogmsh[0.3.1] with fc_tools[0.0.36], fc_bench[0.1.4], fc_amat[0.1.4], ...
    fc_meshtools[0.1.5], fc_graphics4mesh[0.1.7].
Configured to use gmsh 4.13.1 with default MSH file format version 4.1
```

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

```
Using fc_oogmsh[0.3.1] with fc_tools[0.0.36], fc_bench[0.1.4], fc_amat[0.1.4], ...
    fc_meshtools[0.1.5], fc_graphics4mesh[0.1.7].
Configured to use gmsh 4.13.1 with default MSH file format version 4.1
```

For **uninstalling**, one just has to delete directory:

`~/Matlab/fc-oogmsh-full`

3 gmsh interface

All functions provided in this section use `gmsh` to create a mesh file from a `gmsh` geometry script file (extension *.geo*).

3.1 function fc_oogmsh.gmsh.buildmesh2d

This function uses `gmsh` and a *.geo* file (describing a 2D-geometry) to generate a 2D-mesh.

Syntaxe

```
meshfile=fc_oogmsh.gmsh.buildmesh2d(geofile,N)

meshfile=fc_oogmsh.gmsh.buildmesh2d(geofile,N,Name,Value)
```

Description

meshfile=fc_oogmsh.gmsh.buildmesh2d(geofile,N) create a 2D-mesh using `gmsh` and the *geo* file geofile (without path). The integer N has two functions : numbering the name of the generated mesh as <geofile without extension and path> + <-N.msh> and passing this number to `gmsh` via the option "-setnumber N <N>". Usually we used this parameter in `gmsh` to set the prescribed mesh element size at the points. (see given *geo* files)
As output return a file name (with full path) corresponding to the mesh generated by `gmsh`.

meshfile=fc_oogmsh.gmsh.buildmesh2d(geofile,N,Name,Value, ...) specifies function options using one or more Name,Value pair arguments. The Name options can be

- 'geodir' : to specify the directory of the *geo* file geofile ,
- 'meshdir' : to specify the directory where the mesh file will be written,
- 'meshfile' : to specify the name of the mesh file (with path and `.msh` extension),
- 'check' : to perform various consistency checks on mesh with `gmsh`, if Value is true. (default : false)
- 'force' : to force meshing even if the mesh file already exists if Value is true (default : false)
- 'verbose' : to specify the degree of verbosity (0, silence; 2, default; ...)
- 'strings' : cells array of strings corresponding to `gmsh` options given with `-string "..."` (default empty) (see `gmsh` documentation). For example, Value could be:

  `{'Mesh.Algorithm=1;', 'Mesh.ScalingFactor=2;'}`

- `'Loptions'` : cells array of strings, list of options add to `gmsh` command line (default empty) For example, Value could be:

  `{'-setnumber R 1','-setnumber Mesh.Nodes 1 '}`

  and so `gmsh` binary will be executed by

  `gmsh ... -setnumber R 1 -setnumber Mesh.Nodes 1 ... <geofile>`

- 'MshFileVersion' : to specify the MSH file format version. Value could be
  - '2.2' if gmsh version $\geqslant$ 2.16.0,
  - '4.0' if gmsh version $\geqslant$ 4.0.0,
  - '4.1' if gmsh version $\geqslant$ 4.1.0.

Examples All the following examples use the *.geo* file `condenser110C.geo` which is in the directory `geodir` of the toolbox.

Matlab code with output

```
disp('**** fc_oogmsh.gmsh.buildmesh2d : 1st call')
meshfile=fc_oogmsh.gmsh.buildmesh2d('condenser110C',25,'force',true);
disp('**** fc_oogmsh.gmsh.buildmesh2d : 2nd call')
meshfile=fc_oogmsh.gmsh.buildmesh2d('condenser110C',25);
```

```
**** fc_oogmsh.gmsh.buildmesh2d : 1st call
[fc-oogmsh] Input file : <fc-oogmsh>/geodir/2d/condenser110C.geo
[fc-oogmsh] Starting building mesh <fc-oogmsh>/meshes/condenser110C-25.msh with gmsh 4.13.1
[fc-oogmsh] Using command : gmsh -2 -setnumber N 25 -string "Mesh.MshFileVersion=4.1;" <fc-oogmsh>/geodir/2d/condenser110C.geo -o ...
       <fc-oogmsh>/meshes/condenser110C-25.msh
 Be patient...
[fc-oogmsh] Using gmsh 4.13.1 to write MSH file format version 4.1 in <fc-oogmsh>/meshes/condenser110C-25.msh
**** fc_oogmsh.gmsh.buildmesh2d : 2nd call
[fc-oogmsh] Input file : <fc-oogmsh>/geodir/2d/condenser110C.geo
[fc-oogmsh] Mesh file <fc-oogmsh>/meshes/condenser110C-25.msh [version 4.1] already exists.
  -> Use "force" flag to rebuild if needed.
```

Matlab code with output

```
meshfile=fc_oogmsh.gmsh.buildmesh2d('condenser110C',25,'force',true, ...
    'verbose',4, 'strings',{'Mesh.Algorithm=1;', 'Mesh.ScalingFactor=2;'});
```

```
[fc-oogmsh] Input file : <fc-oogmsh>/geodir/2d/condenser110C.geo
[fc-oogmsh] Overwritting mesh file <fc-oogmsh>/meshes/condenser110C-25.msh
[fc-oogmsh] Starting building mesh <fc-oogmsh>/meshes/condenser110C-25.msh with gmsh 4.13.1
[fc-oogmsh] Using command : '/home/cuvelier/bin/gmsh' -2 -setnumber N 25 -string "Mesh.Algorithm=1;Mesh.ScalingFactor=2;Mesh.MshFileVersion=4.1;" ...
       '/home/cuvelier/Travail/Recherch/Matlab/fc-config/build/tmpdir/packages/fc_oogmsh-0.3.1/geodir/2d/condenser110C.geo' -o ...
       '/home/cuvelier/Travail/Recherch/Matlab/fc-config/build/tmpdir/packages/fc_oogmsh-0.3.1/meshes/condenser110C-25.msh'
 Be patient...
[fc-oogmsh] Starting building mesh <fc-oogmsh>/meshes/condenser110C-25.msh with gmsh 4.13.1
[fc-oogmsh] Using command : gmsh -2 -setnumber N 25 -string "Mesh.Algorithm=1;Mesh.ScalingFactor=2;Mesh.MshFileVersion=4.1;" ...
       <fc-oogmsh>/geodir/2d/condenser110C.geo -o <fc-oogmsh>/meshes/condenser110C-25.msh
 Be patient...
[fc-oogmsh] gmsh output :
Info    : Running '/home/cuvelier/bin/gmsh -2 -setnumber N 25 -string Mesh.Algorithm=1;Mesh.ScalingFactor=2;Mesh.MshFileVersion=4.1; ...
       <fc-oogmsh>/geodir/2d/condenser110C.geo -o <fc-oogmsh>/meshes/condenser110C-25.msh' [Gmsh 4.13.1, 1 node, max. 1 thread]
Info    : Started on Tue Jan 28 10:51:40 2025
Info    : Reading '<fc-oogmsh>/geodir/2d/condenser110C.geo'...
Info    : [  0%] Difference
Info    : [ 10%] Difference
Info    : [ 20%] Difference
Info    : [ 30%] Difference - Performing Face-Face intersection
Info    : [ 70%] Difference - Filling splits of edges
Info    : [ 80%] Difference - Making faces
Info    : [ 90%] Difference - Adding holes
Info    : [  0%] Difference
Info    : [ 10%] Difference
Info    : [ 20%] Difference
Info    : [ 30%] Difference - Performing Face-Face intersection
Info    : [ 70%] Difference
Info    : [ 80%] Difference - Making faces
Info    : [ 90%] Difference - Adding holes

Info    : Done reading '<fc-oogmsh>/geodir/2d/condenser110C.geo'
Info    : Meshing 1D...
Info    : [  0%] Meshing curve 10 (Ellipse)
Info    : [ 10%] Meshing curve 11 (Ellipse)
Info    : [ 20%] Meshing curve 12 (Ellipse)
Info    : [ 30%] Meshing curve 13 (Ellipse)
Info    : [ 40%] Meshing curve 14 (Line)
Info    : [ 40%] Meshing curve 15 (Line)
Info    : [ 50%] Meshing curve 16 (Line)
Info    : [ 60%] Meshing curve 17 (Line)
Info    : [ 70%] Meshing curve 18 (Ellipse)
Info    : [ 70%] Meshing curve 19 (Ellipse)
Info    : [ 80%] Meshing curve 20 (Ellipse)
Info    : [ 90%] Meshing curve 21 (Ellipse)
Info    : [100%] Meshing curve 22 (Ellipse)
Info    : Done meshing 1D (Wall 0.00126491s, CPU 0.004585s)
Info    : Meshing 2D...
Info    : [  0%] Meshing surface 11 (Plane, Frontal-Delaunay)
Info    : [ 30%] Meshing surface 13 (Plane, Frontal-Delaunay)
Info    : [ 50%] Meshing surface 15 (Plane, Frontal-Delaunay)
Info    : [ 70%] Meshing surface 17 (Plane, Frontal-Delaunay)
Info    : [ 90%] Meshing surface 101 (Plane, Frontal-Delaunay)
Info    : Done meshing 2D (Wall 0.0539134s, CPU 0.189045s)
Info    : 2069 nodes 4211 elements
Info    : Writing '<fc-oogmsh>/meshes/condenser110C-25.msh'...
Info    : Done writing '<fc-oogmsh>/meshes/condenser110C-25.msh'
Info    : Stopped on Tue Jan 28 10:51:40 2025 (From start: Wall 0.0713624s, CPU 0.244836s)

[fc-oogmsh] Using gmsh 4.13.1 to write MSH file format version 4.1 in <fc-oogmsh>/meshes/condenser110C-25.msh
```

3.2 function `fc_oogmsh.gmsh.buildmesh3d`

This function uses `gmsh` and a *.geo* file (describing a 3D-geometry) to generate a 3D-mesh. See function `gmsh.buildmesh2d` for usage and options (section 3.1).

3.3 function fc_oogmsh.gmsh.buildmesh3ds

This function uses **gmsh** and a *.geo* file (describing a 3D surface geometry or a 3D-geometry) to generate a 3D surface mesh. See function **gmsh.buildmesh2d** for usage and options (section 3.1).

3.4 function fc_oogmsh.gmsh.buildpartmesh2d

This function uses **gmsh** and a *.msh* file (containing a 2D-mesh) to generate a 2D partioned mesh.

Syntaxe

```
partmeshfile=fc_oogmsh.gmsh.buildpartmesh2d(meshfile,np)

partmeshfile=fc_oogmsh.gmsh.buildpartmesh2d(meshfile,np,Name,Value)
```

Description

`partmeshfile=fc_oogmsh.gmsh.buildpartmesh2d(meshfile,np)` create a 2D partitioned mesh using **gmsh** and the *msh* file meshfile (with path). The integer np is the number of partitions.
As output return a file name (with full path) corresponding to the partitioned mesh generated by **gmsh**. The output file name is construct as following : <meshfile without extension>-part<np>.msh

`partmeshfile=fc_oogmsh.gmsh.buildpartmesh2d(meshfile,np,Name,Value, ...)` specifies function options using one or more Name,Value pair arguments. The Name options can be

- 'savedir' : to specify the directory where the partitioned mesh file will be written,
- 'check' : to perform various consistency checks on mesh with **gmsh**, if Value is true. (default : false)
- 'force' : to force meshing even if the mesh file already exists if Value is true (default : false)
- 'verbose' : to specify the degree of verbosity (0, silence; 2, default; ...)
- 'strings' : cells array of strings corresponding to **gmsh** options given with `-string "..."` (default empty) (see **gmsh** documentation)
- 'MshFileVersion' : to specify the MSH file format version. Value could be
  - '2.2' if gmsh version $\geqslant$ 2.16.0,
  - '4.0' if gmsh version $\geqslant$ 4.0.0,
  - '4.1' if gmsh version $\geqslant$ 4.1.0.

Examples

Matlab code with output

```
meshfile=fc_oogmsh.gmsh.buildmesh2d('condenser110C',25,'verbose',0);
pmfile=fc_oogmsh.gmsh.buildpartmesh2d(meshfile,5,'force',true);
```

```
[fc-oogmsh] Input file : <fc-oogmsh>/meshes/condenser110C-25.msh
[fc-oogmsh] Starting building mesh <fc-oogmsh>/meshes/condenser110C-25-part5.msh with gmsh 4.13.1
[fc-oogmsh] Using command : gmsh -2 -part 5 -string "Mesh.MshFileVersion=4.1;" -saveall <fc-oogmsh>/meshes/condenser110C-25.msh -o ...
       <fc-oogmsh>/meshes/condenser110C-25-part5.msh
 Be patient...
[fc-oogmsh] Using gmsh 4.13.1 to write MSH file format version 4.1 in <fc-oogmsh>/meshes/condenser110C-25-part5.msh
```

Matlab code with output

```
meshfile=fc_oogmsh.gmsh.buildmesh2d('condenser110C',25,'verbose',0);
pmfile=fc_oogmsh.gmsh.buildpartmesh2d(meshfile,5,'force',true, 'verbose',4, ...
    'strings',{'Mesh.MetisAlgorithm=3;'});
```

```
[fc-oogmsh] Input file : <fc-oogmsh>/meshes/condenser110C-25.msh
[fc-oogmsh] Overwritting mesh file <fc-oogmsh>/meshes/condenser110C-25-part5.msh
[fc-oogmsh] Starting building mesh <fc-oogmsh>/meshes/condenser110C-25-part5.msh with gmsh 4.13.1
[fc-oogmsh] Using command : '/home/cuvelier/bin/gmsh' -2 -part 5 -string "Mesh.MetisAlgorithm=3;Mesh.MshFileVersion=4.1;" -saveall ...
       '/home/cuvelier/Travail/Recherch/Matlab/fc-config/build/tmpdir/packages/fc_oogmsh-0.3.1/meshes/condenser110C-25.msh' -o ...
       '/home/cuvelier/Travail/Recherch/Matlab/fc-config/build/tmpdir/packages/fc_oogmsh-0.3.1/meshes/condenser110C-25-part5.msh'
 Be patient...
[fc-oogmsh] Starting building mesh <fc-oogmsh>/meshes/condenser110C-25-part5.msh with gmsh 4.13.1
[fc-oogmsh] Using command : gmsh -2 -part 5 -string "Mesh.MetisAlgorithm=3;Mesh.MshFileVersion=4.1;" -saveall <fc-oogmsh>/meshes/condenser110C-25.msh -o ...
       <fc-oogmsh>/meshes/condenser110C-25-part5.msh
 Be patient...
[fc-oogmsh] gmsh output :
Info    : Running '/home/cuvelier/bin/gmsh -2 -part 5 -string Mesh.MetisAlgorithm=3;Mesh.MshFileVersion=4.1; -saveall <fc-oogmsh>/meshes/condenser110C-25.msh ...
       -o <fc-oogmsh>/meshes/condenser110C-25-part5.msh' [Gmsh 4.13.1, 1 node, max. 1 thread]
Info    : Started on Tue Jan 28 10:51:59 2025
Info    : Reading '<fc-oogmsh>/meshes/condenser110C-25.msh'...
Info    : 31 entities
Info    : 2069 nodes
Info    : 4146 elements
Info    : Done reading '<fc-oogmsh>/meshes/condenser110C-25.msh'
Info    : Meshing 1D...
Info    : Done meshing 1D (Wall 3.8571e-05s, CPU 7.9e-05s)
Info    : Meshing 2D...
Info    : Done meshing 2D (Wall 2.702e-05s, CPU 5.5e-05s)
Info    : 2069 nodes 4159 elements
Info    : Partitioning mesh...
Info    : Running METIS with npart:5, sizeof(idx_t):64, ptype:default, ufactor:default, ctype:shem, rtype:greedy, objtype:cut, minconn:default
Info    : 5 partitions, 125 total edge-cuts
Info    : Done partitioning mesh (Wall 0.00785511s, CPU 0.018703s)
Info    :  - Repartition of 13 points: 1(min) 3(max) 2.6(avg)
Info    :  - Repartition of 237 lines: 42(min) 52(max) 47.4(avg)
Info    :  - Repartition of 3909 triangles: 781(min) 783(max) 781.8(avg)
Info    : Creating partition topology...
Info    :  - Creating partition curves
Info    :  - Creating partition points
Info    : Done creating partition topology (Wall 0.0012486s, CPU 0.00722s)
Info    : Writing '<fc-oogmsh>/meshes/condenser110C-25-part5.msh'...
Info    : Done writing '<fc-oogmsh>/meshes/condenser110C-25-part5.msh'
Info    : Stopped on Tue Jan 28 10:51:59 2025 (From start: Wall 0.0236109s, CPU 0.067347s)

[fc-oogmsh] Using gmsh 4.13.1 to write MSH file format version 4.1 in <fc-oogmsh>/meshes/condenser110C-25-part5.msh
```

3.5 function fc_oogmsh.gmsh.buildpartmesh3d

This function uses `gmsh` and a *.msh* file (containing of a 3D-mesh) to generate a 3D partioned mesh. See function `gmsh.buildpartmesh2d` for usage and options (section 3.4).

3.6 function fc_oogmsh.gmsh.buildpartmesh3ds

This function uses `gmsh` and a *.msh* file (containing of a 3D surface mesh) to generate a 3D partioned surface mesh. See function `gmsh.buildpartmesh2d` for usage and options (section 3.4).

3.7 function fc_oogmsh.gmsh.buildPartRectangle

This function uses `gmsh` and the *geodir/rectanglepart.geo* file to generate a 2D regular partioned mesh of the rectangle $[0, Lx] \times [0, Ly]$ with $Nx \times Ny$ partitions.

Syntaxe

```
meshfile=fc_oogmsh.gmsh.buildpartrectangle(Lx,Ly,Nx,Ny,N)

meshfile=fc_oogmsh.gmsh.buildpartrectangle(Lx,Ly,Nx,Ny,N, Name,Value)
```

Description

`meshfile=fc_oogmsh.gmsh.buildpartrectangle(Lx,Ly,Nx,Ny,N)` create a 2D regular partitioned mesh using `gmsh` of the rectangle $[0, Lx] \times [0, Ly]$ with $Nx \times Ny$ partitions. The N parameter is passed to `gmsh` to set the prescribed mesh element size at the points

As output return a file name (with full path) corresponding to the partitioned mesh generated by `gmsh`. The default output file name is construct as following :

```
sprintf('rectanglepart_Lx%.3f_Ly%.3f_Nx%d_Ny%d_N%d.msh',Lx,Ly,Nx,Ny,N)
```

meshfile=fc_oogmsh.gmsh.buildpartrectangle(Lx,Ly,Nx,Ny,N,Name,Value, ...) specifies function options using one or more Name,Value pair arguments (see the fc_oogmsh.gmsh.buildmesh2d, section 3.1).

Examples All the following examples ...

Matlab code with output

```
pmfile=fc_oogmsh.gmsh.buildpartrectangle(1,1,3,2,100,'force',true);
```

```
[fc-oogmsh] Input file : <fc-oogmsh>/geodir/2d/rectanglepart.geo
[fc-oogmsh] Starting building mesh <fc-oogmsh>/meshes/rectanglepart-Lx1.000-Ly1.000-Nx3-Ny2-N100.msh with gmsh 4.13.1
[fc-oogmsh] Using command : gmsh -2 -string "Mesh.MshFileVersion=4.1;" -setnumber N 100 -setnumber NX 3 -setnumber NY 2 -setnumber LX 1 -setnumber LY 1 ...
       <fc-oogmsh>/geodir/2d/rectanglepart.geo -o <fc-oogmsh>/meshes/rectanglepart-Lx1.000-Ly1.000-Nx3-Ny2-N100.msh
 Be patient...
[fc-oogmsh] Using gmsh 4.13.1 to write MSH file format version 4.1 in <fc-oogmsh>/meshes/rectanglepart-Lx1.000-Ly1.000-Nx3-Ny2-N100.msh
```

Matlab code with output

```
pmfile=fc_oogmsh.gmsh.buildpartrectangle(1,1,3,2,100,'verbose',4, ...
    'force',true,'meshfile','./toto.msh');;
```

```
[fc-oogmsh] Input file : <fc-oogmsh>/geodir/2d/rectanglepart.geo
[fc-oogmsh] Starting building mesh ./toto.msh with gmsh 4.13.1
[fc-oogmsh] Using command : '/home/cuvelier/bin/gmsh' -2 -string "Mesh.MshFileVersion=4.1;" -setnumber N 100 -setnumber NX 3 -setnumber NY 2 -setnumber LX 1 ...
       -setnumber LY 1 '/home/cuvelier/Travail/Recherch/Matlab/fc-config/build/tmpdir/packages/fc_oogmsh-0.3.1/geodir/2d/rectanglepart.geo' -o './toto.msh'
 Be patient...
[fc-oogmsh] Starting building mesh ./toto.msh with gmsh 4.13.1
[fc-oogmsh] Using command : gmsh -2 -string "Mesh.MshFileVersion=4.1;" -setnumber N 100 -setnumber NX 3 -setnumber NY 2 -setnumber LX 1 -setnumber LY 1 ...
       <fc-oogmsh>/geodir/2d/rectanglepart.geo -o ./toto.msh
 Be patient...
[fc-oogmsh] gmsh output :
Info    : Running '/home/cuvelier/bin/gmsh -2 -string Mesh.MshFileVersion=4.1; -setnumber N 100 -setnumber NX 3 -setnumber NY 2 -setnumber LX 1 -setnumber LY 1 ...
       <fc-oogmsh>/geodir/2d/rectanglepart.geo -o ./toto.msh' [Gmsh 4.13.1, 1 node, max. 1 thread]
Info    : Started on Tue Jan 28 10:52:17 2025
Info    : Reading '<fc-oogmsh>/geodir/2d/rectanglepart.geo'...
Info    : Reading '<fc-oogmsh>/geodir/2d/partitions01_data.geo'...
Info    : Done reading '<fc-oogmsh>/geodir/2d/partitions01_data.geo'
Info    : Reading '<fc-oogmsh>/geodir/2d/partitions_shape.geo'...
Info    : Done reading '<fc-oogmsh>/geodir/2d/partitions_shape.geo'
Info    : Done reading '<fc-oogmsh>/geodir/2d/rectanglepart.geo'
Info    : Meshing 1D...
Info    : [  0%] Meshing curve 1 (Line)
Info    : [ 10%] Meshing curve 2 (Line)
Info    : [ 20%] Meshing curve 3 (Line)
Info    : [ 20%] Meshing curve 4 (Line)
Info    : [ 30%] Meshing curve 5 (Line)
Info    : [ 30%] Meshing curve 6 (Line)
Info    : [ 40%] Meshing curve 7 (Line)
Info    : [ 50%] Meshing curve 8 (Line)
Info    : [ 50%] Meshing curve 9 (Line)
Info    : [ 60%] Meshing curve 10 (Line)
Info    : [ 60%] Meshing curve 11 (Line)
Info    : [ 70%] Meshing curve 12 (Line)
Info    : [ 80%] Meshing curve 13 (Line)
Info    : [ 80%] Meshing curve 14 (Line)
Info    : [ 90%] Meshing curve 15 (Line)
Info    : [ 90%] Meshing curve 16 (Line)
Info    : [100%] Meshing curve 17 (Line)
Info    : Done meshing 1D (Wall 0.00107747s, CPU 0.002002s)
Info    : Meshing 2D...
Info    : [  0%] Meshing surface 19 (Plane, Frontal-Delaunay)
Info    : [ 20%] Meshing surface 21 (Plane, Frontal-Delaunay)
Info    : [ 40%] Meshing surface 23 (Plane, Frontal-Delaunay)
Info    : [ 60%] Meshing surface 25 (Plane, Frontal-Delaunay)
Info    : [ 70%] Meshing surface 27 (Plane, Frontal-Delaunay)
Info    : [ 90%] Meshing surface 29 (Plane, Frontal-Delaunay)
Info    : Done meshing 2D (Wall 0.196988s, CPU 0.367492s)
Info    : 12087 nodes 24486 elements
Info    : Writing './toto.msh'...
Info    : Done writing './toto.msh'
Info    : Stopped on Tue Jan 28 10:52:17 2025 (From start: Wall 0.218647s, CPU 0.399236s)

[fc-oogmsh] Using gmsh 4.13.1 to write MSH file format version 4.1 in ./toto.msh
```

4 ooGmsh4 class (version 4.x)

The ooGmsh4 class can be used to read `gmsh` mesh files with the MSH ASCII file format version `4.1` since `gmsh 4.1.0` ([4], section 9.1) or version `4.0` since `gmsh 4.0.0`.

The `gmsh`'s native `"MSH"` file format (version 4.x) is used to store meshes and associated post-processing datasets either save as an ASCII file or a binary file with extension `.msh`. The focus of the ooGmsh4 class is to read only meshes contained in an ASCII file. Currently, it is not planned to read post-processing datasets.

As described in [4], section 9.1: *the MSH file format version 4 (current revision: version 4.1) contains one mandatory section giving information about the file (`$MeshFormat`), followed by several optional sections defining the physical group names (`$PhysicalName`), the elementary geometrical enti-*

ties (`$Entities`*), the partitioned entities (*`$PartitionedEntities`*), the nodes* `$Nodes`*), the elements (*`$Elements`*), the periodicity relations (*`$Periodic`*), the ghost* elements (`$GhostElements`) and the post-processing datasets (`$NodeData`, `$ElementData`, `$ElementNodeData`).

For each section, the ooGmsh4 class has a property with corresponding name. The properties of this class are:

Properties of ooGmsh4 class

dim	:	space dimension (2 or 3)
nq	:	number of nodes/vertices.
q	:	nodes/vertices array with dimension dim-by-nq.
toGlobal	:	...
MeshFormat	:	structure
PhysicalNames	:	(optional), array of PhysicalName structure
Entities	:	structure
PartitionedEntities	:	(optional) structure
Nodes	:	structure
Elements	:	structure
PeriodicLinks	:	(optional), array of PeriodicLink structure

The structures MeshFormat, PhysicalNames, Entities, PartitionedEntities, Nodes, Elements and PeriodicLinks are described in section 4.2. In the following subsections, Gh is an ooGmsh4 object.

4.1 Methods

4.1.1 ooGms4 constructor

The ooGmsh4 class have only one constructor :

```
Gh=fc_oogmsh.ooGmsh4(meshfile)
Gh=fc_oogmsh.ooGmsh4(meshfile, 'verbosity', Value)
```

where meshfile is the name of ... a mesh file. The 'verbosity' Key/Value option can be used to print some informations, when reading the file meshfile, if Value is true. Default is false

Matlab code with output

```
fprintf('1) Building the mesh\n')
meshfile=fc_oogmsh.gmsh.buildmesh2d('condenser',10, ...
    'verbose',0,'force',true);
fprintf('2) Reading the mesh\n');
Gh = fc_oogmsh.ooGmsh4(meshfile,'verbose',true);
fprintf('-> Gh is an ooGmsh4 object containing a MSH file version ...
    %s\n',Gh.MeshFormat.version)
fprintf('3) Displaying Gh\n');
Gh
```

```
1) Building the mesh
2) Reading the mesh
Optional string "$PhysicalNames" not found
Reading $Entities section seem OK
Optional string "$PartitionedEntities" not found
Reading $Nodes section seem OK
Reading $Elements section seem OK
Optional string "$Periodic" not found
-> Gh is an ooGmsh4 object containing a MSH file version 4.1
3) Displaying Gh

Gh =

  fc_oogmsh.ooGmsh4 with properties:
                    q: (2x8162 double)
                   nq: 8162 double
                  dim: 2 double
                    d: 2 double
             toGlobal: (1x8162 double)
             meshfile: (1x110 char)
      partitionnedfile: 0 logical
           MeshFormat: (1x1 struct)
        PhysicalNames: []
             Entities: (1x1 struct)
    PartitionedEntities: []
                Nodes: (1x1 struct)
             Elements: (1x1 struct)
         PeriodicLinks: []
                 Info: (1x1 struct)
               orders: 1 double
```

4.1.2 `info` method

```
info(Gh)
Gh.info()
Gh.info(Key, Value, ...)
```

Description

Gh.**info**()

print informations on class fields with 3 levels of recursivity (i.e. field of field of field).

Gh.**info**(Key, Value, ...)

specifies function options using one or more Key,Value pair arguments. The Key options can be

- 'maxlevel' : level of recursivity, default is 3.
- 'tab' : number of space characters between two levels of recursivity, default is 4.

Matlab code with output

```
meshfile=fc_oogmsh.gmsh.buildmesh2d('condenser',6, 'verbose',0,'force',true);
Gh = fc_oogmsh.ooGmsh4(meshfile);
Gh.info('maxlevel',2);
```

```
fc_oogmsh.ooGmsh4 with properties:
  [1] q : [2 3036] double
  [1] nq : [1 1] double
  [1] dim : [1 1] double
  [1] d : [1 1] double
  [1] toGlobal : [1 3036] double
  [1] meshfile : [1 109] char
  [1] partitionnedfile : [1 1] logical
  [1] MeshFormat : [1 1] struct
      [2] version : [1 3] char
      [2] file_type : [1 1] double
      [2] data_size : [1 1] double
  [1] PhysicalNames : [0 0] double
  [1] Entities : [1 1] struct
      [2] numPoints : [1 1] double
      [2] Points : [1 11] struct
      [2] numCurves : [1 1] double
      [2] Curves : [1 10] struct
      [2] numSurfaces : [1 1] double
      [2] Surfaces : [1 1] struct
      [2] numVolumes : [1 1] double
      [2] Volumes : [1 0] struct
  [1] PartitionedEntities : [0 0] double
  [1] Nodes : [1 1] struct
      [2] numEntityBlocks : [1 1] double
      [2] numNodes : [1 1] double
      [2] minNodeTag : [1 1] double
      [2] maxNodeTag : [1 1] double
      [2] EntityBlocks : [1 21] struct
  [1] Elements : [1 1] struct
      [2] numEntityBlocks : [1 1] double
      [2] numElements : [1 1] double
      [2] minElementTag : [1 1] double
      [2] maxElementTag : [1 1] double
      [2] EntityBlocks : [1 19] struct
      [2] ElementTypes : [1 19] double
  [1] PeriodicLinks : [0 0] double
  [1] Info : [1 1] struct
      [2] meshfile : [1 109] char
  [1] orders : [1 1] double
```

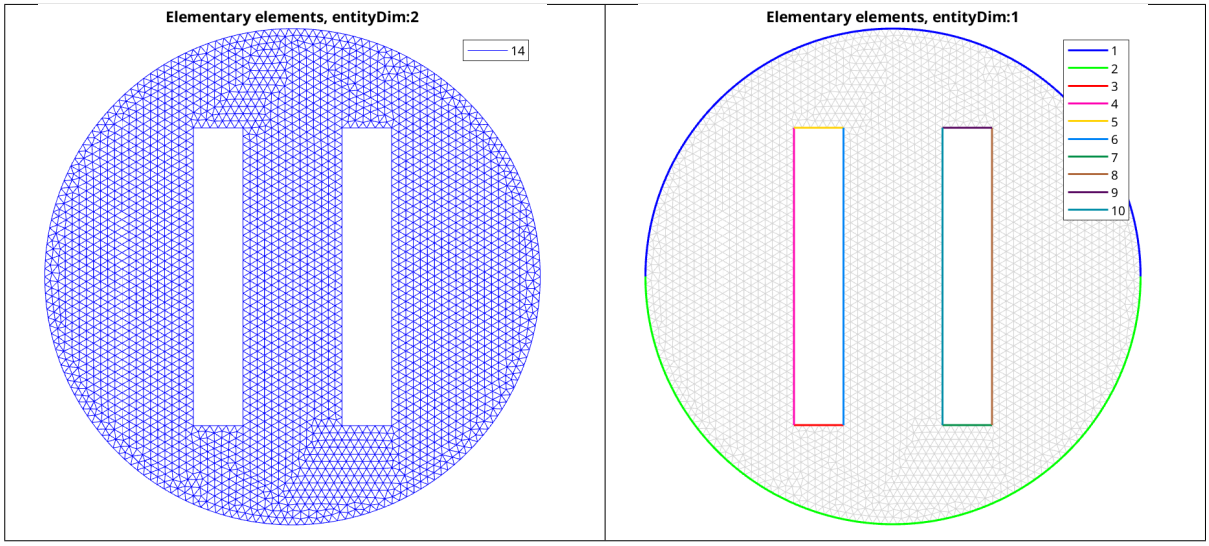


Figure 1: *Elementary Tag* elements of the *geofile* `condenser.geo`

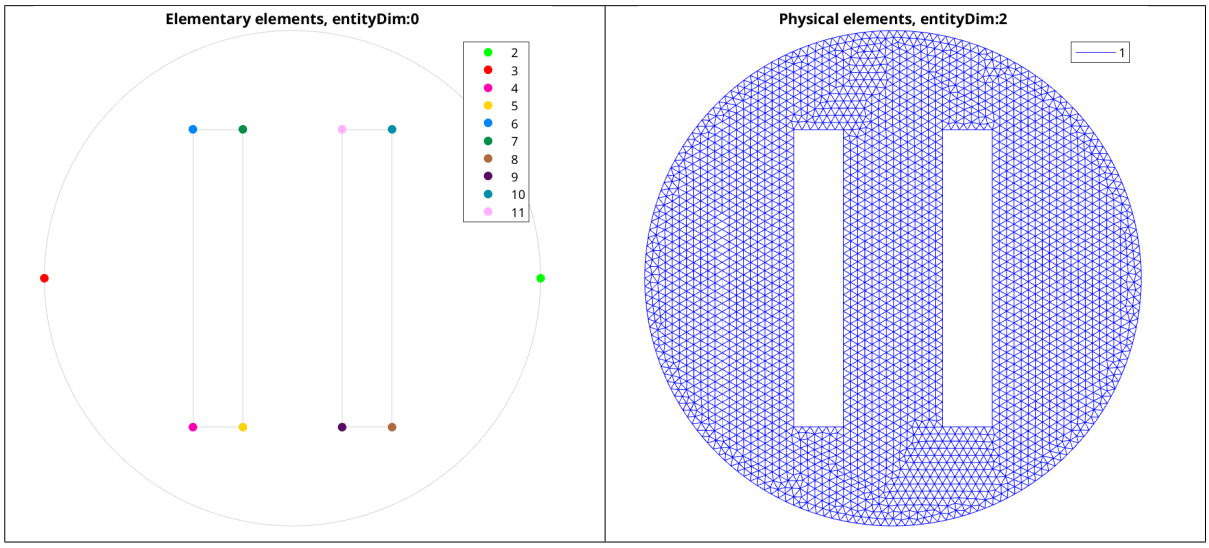


Figure 2: *Physical Tag* elements of the *geofile* `condenser.geo`

In the *geofile* `condenser.geo` the *Physical Tags* are created from the *Elementary Tags* as follow

```
...
Physical Line(1) = {1, 2};
Physical Line(98) = {5, 6, 3, 4};
Physical Line(99) = {9, 8, 7, 10};
Physical Surface(1) = {14};
```

4.1.3 get_ElementaryTags method

```
eltags=get_ElementaryTags(Gh,EltType)
eltags=Gh.get_ElementaryTags(EltType)
```

Description

`eltags=Gh.get_ElementaryTags(EltType)`

returns all the elementary tags associated with elements of type EltType as an array with unique elements. EltType is described in section **??**. For example, EltType is 1 for `2-nodes line` (i.e 1-simplex of order 1), EltType is 2 for `3-nodes triangle` (i.e 2-simplex of order 1) and EltType is 4 for `4-nodes tetrahedron` (i.e 3-simplex of order 1).

Matlab code with output

```
eltags1=Gh.get_ElementaryTags(1)
eltags2=Gh.get_ElementaryTags(2)
```

```
eltags1 =

    1    2    3    4    5    6    7    8    9   10


eltags2 =

   14
```

4.1.4 get_PhysicalTags method

```
phtags=get_PhysicalTags(Gh,EltType)
phtags=Gh.get_PhysicalTags(EltType)
```

Description

phtags=Gh.get_PhysicalTags(EltType)

returns all the elementary tags associated with elements of type EltType as an array with unique elements.

Matlab code with output

```
phtags1=Gh.get_PhysicalTags(1)
phtags2=Gh.get_PhysicalTags(2)
```

```
phtags1 =

    1   98   99

phtags2 =

    1
```

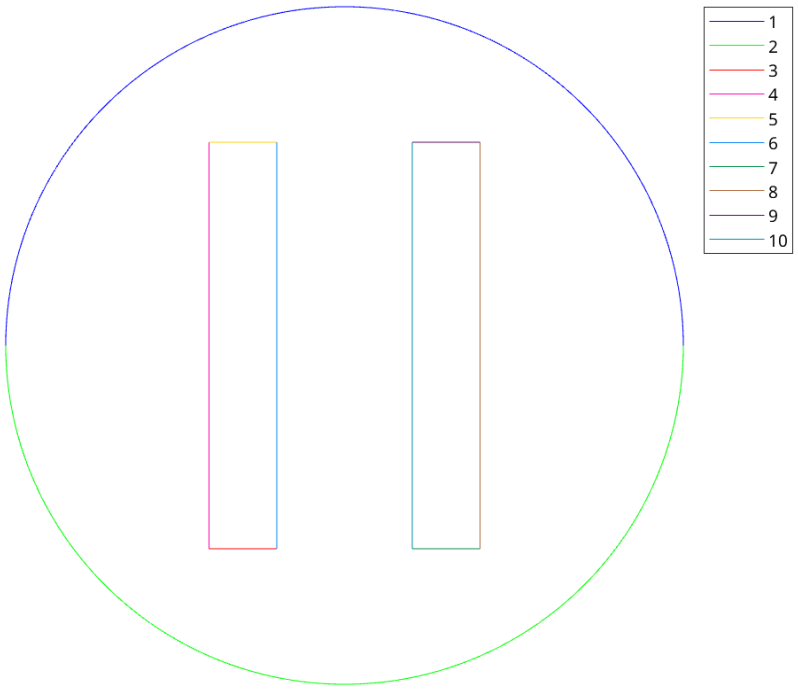
4.1.5 get_me_ElementaryTag method

```
me=get_me_ElementaryTag(Gh,EltType,EltTag)
me=Gh.get_me_ElementaryTag(EltType,EltTag)
```

Description

me=Gh.get_me_ElementaryTag(EltType,EltTag)

returns me the connectivity array of mesh elements of type and *elementary tag* given respectively by EltType and EltTag. This array is associated with the Gh.q nodes/vertices array.




```
meshfile=fc_oogmsh.gmsh.buildmesh2d('condenser',6, 'verbose',0);
Gh = fc_oogmsh.ooGmsh4(meshfile);
eltags1=Gh.get_ElementaryTags(1);
n1=length(eltags1);
colors = fc_tools.graphics.selectColors(length(eltags1));
figure(1)
hold on
for i=1:n1
  me=Gh.get_me_ElementaryTag(1,eltags1(i));
  h(i)=fc_graphics4mesh.plotmesh(Gh.q,me,'color',colors(i,:));
  clegend{i}=num2str(eltags1(i));
end
legend(h,clegend,'Location','NorthEastOutside','AutoUpdate','off');
axis image;axis off
```

Listing 1: Plot curves mesh elements by using get_me_ElementaryTags function

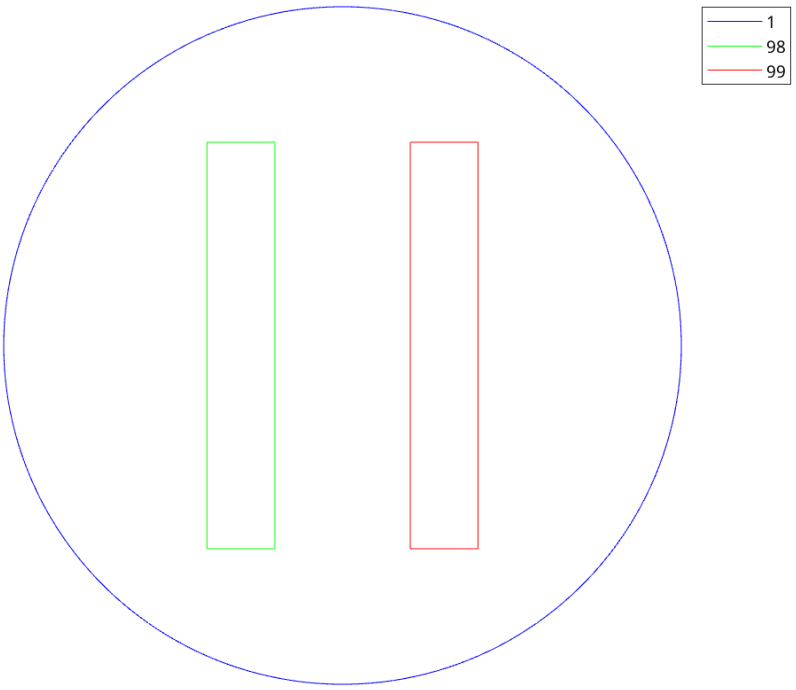
4.1.6 get_me_PhysicalTag method

```
me=get_me_PhysicalTag(Gh,EltType,PhyTag)
me=Gh.get_me_PhysicalTag(EltType,PhyTag)
```

Description

get_me_PhysicalTag(Gh,EltType,PhyTag)

returns me the connectivity array of mesh elements of type and *physical tag* given respectively by EltType and PhyTag. This array is associated with the Gh.q nodes/vertices array.




```
meshfile=fc_oogmsh.gmsh.buildmesh2d('condenser',6, 'verbose',0);
Gh = fc_oogmsh.ooGmsh4(meshfile);
phtags1=Gh.get_PhysicalTags(1);
n1=length(phtags1);
colors = fc_tools.graphics.selectColors(length(phtags1));
figure(1)
hold on
for i=1:n1
  me=Gh.get_me_PhysicalTag(1,phtags1(i));
  h(i)=fc_graphics4mesh.plotmesh(Gh.q,me,'color',colors(i,:));
  clegend{i}=num2str(phtags1(i));
end
legend(h,clegend,'Location','NorthEastOutside','AutoUpdate','off');
axis image;axis off
```

Listing 2: Plot curves mesh elements by using get_me_PhysicalTags function

4.1.7 get_localmesh_ElementaryTag method

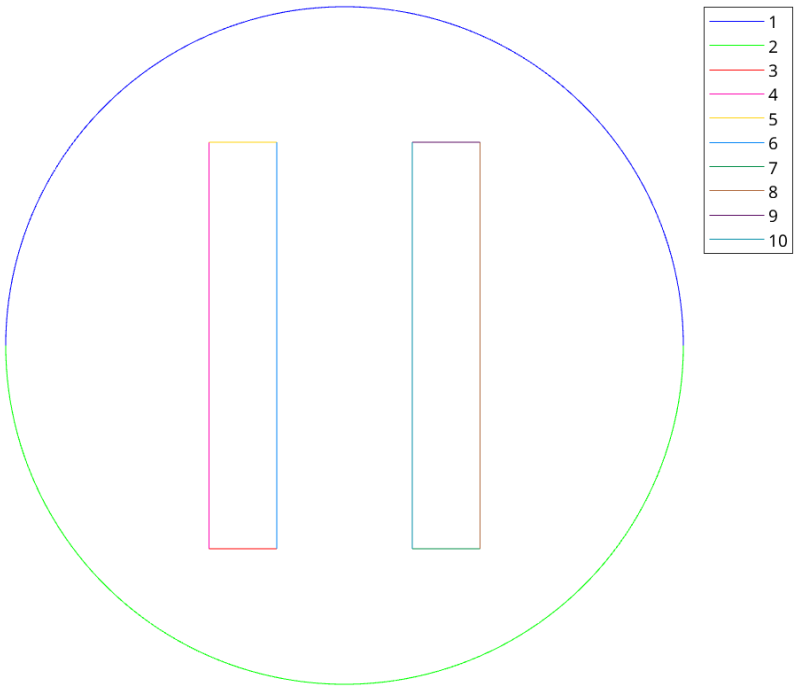
```
[q,me]=Gh.get_localmesh_ElementaryTag(EltType,EltTag)
[q,me,toGlobal]=Gh.get_localmesh_ElementaryTag(EltType,EltTag)
```

[q,me]=Gh.get_localmesh_ElementaryTag(EltType,EltTag)

returns the *local* nodes/vertices array q and the *local* connectivity array me of the element of type EltType and with *elementary tag* given by EltTag.

[q,me,toGlobal]=Gh.get_localmesh_ElementaryTag(EltType,EltTag)

Also returns the *global* tags array toGlobal such that Gh.q(:,toGlobal) is equal to q.




```
meshfile=fc_oogmsh.gmsh.buildmesh2d('condenser',6, 'verbose',0);
Gh = fc_oogmsh.ooGmsh4(meshfile);
eltags1=Gh.get_ElementaryTags(1); % 1: 2-nodes line
n1=length(eltags1);
colors = fc_tools.graphics.selectColors(length(eltags1));
figure(1)
hold on
for i=1:n1
  [q,me]=Gh.get_localmesh_ElementaryTag(1,eltags1(i));
  h(i)=fc_graphics4mesh.plotmesh(q,me,'color',colors(i,:));
  clegend{i}=num2str(eltags1(i));
end
legend(h,clegend,'Location','NorthEastOutside','AutoUpdate','off');
axis image;axis off
```

Listing 3: Plot *2-nodes line* mesh elements by using get_localmesh_ElementaryTag function

4.1.8 get_localmesh_PhysicalTag method

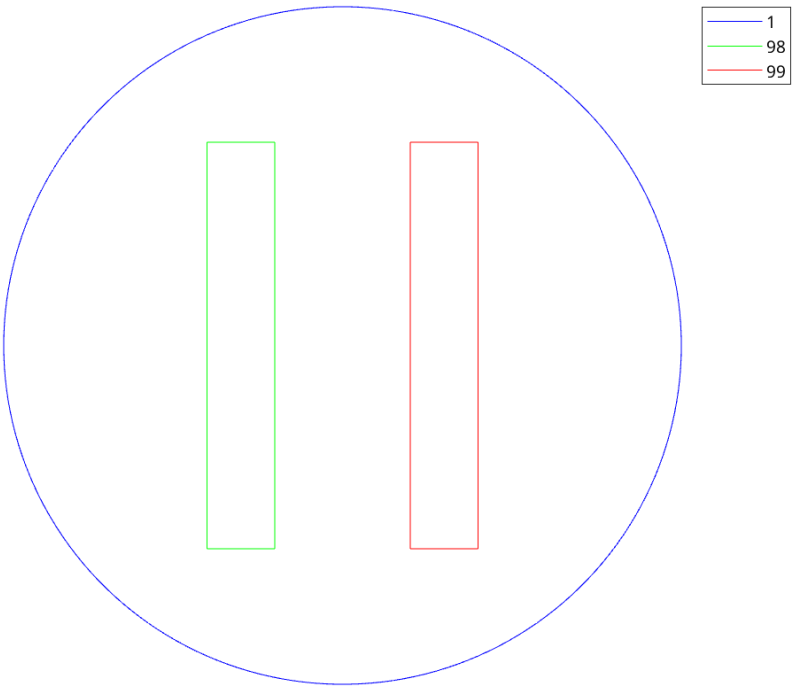
```
[q,me]=Gh.get_localmesh_PhysicalTag(EltType,PhyTag)
[q,me,toGlobal]=Gh.get_localmesh_PhysicalTag(EltType,PhysicalTag)
```

`[q,me]=Gh.get_localmesh_PhysicalTag(EltType,PhyTag)`

returns the *local* nodes/vertices array q and the *local* connectivity array me of the elements of type EltType and with *PhyTag* given by PhysicalTag.

`[q,me,toGlobal]=Gh.get_localmesh_PhysicalTag(EltType,PhyTag)`

Also returns the *global* tags array toGlobal such that Gh.q(:,toGlobal) is equal to q.




```
meshfile=fc_oogmsh.gmsh.buildmesh2d('condenser',6, 'verbose',0);
Gh = fc_oogmsh.ooGmsh4(meshfile);
phtags1=Gh.get_PhysicalTags(1); % 1: 2-nodes line
n1=length(phtags1);
colors = fc_tools.graphics.selectColors(length(phtags1));
figure(1)
hold on
for i=1:n1
  [q,me]=Gh.get_localmesh_PhysicalTag(1,phtags1(i));
  h(i)=fc_graphics4mesh.plotmesh(q,me,'color',colors(i,:));
  clegend{i}=num2str(phtags1(i));
end
legend(h,clegend,'Location','NorthEastOutside','AutoUpdate','off');
axis image;axis off
```

Listing 4: Plot *2-nodes line* mesh elements by using get_localmesh_PhysicalTag function

4.2 Description of properties

Fields of MeshFormat structure

version	:	string, version of the mesh file format.
file_type	:	integer, 0 for ASCII mode, 1 for binary mode.
data_size	:	integer, `sizeof(size_t)`

Fields of the (optional) PhysicalName structure

dimension	:	integer.
physicalTag	:	integer.
name	:	string

Fields of the Entities structure

numPoints	:	integer.
Points	:	array of Point structure.
numCurves	:	integer.
Curves	:	array of Curve structure.
numSurfaces	:	integer.
Surfaces	:	array of Surface structure.
numVolumes	:	integer.
Volumes	:	array of Volume structure.

Fields of (optional) PartitionedEntities structure

numPartitions	:	integer.
numGhostEntities	:	integer.
GhostEntities	:	array of structure.
numPoints	:	integer
Points	:	array of structure.
numCurves	:	integer
Curves	:	array of structure.
numSurfaces	:	integer
Surfaces	:	array of structure.
numVolumes	:	integer.
Volumes	:	array of structure.

Fields of Nodes structure

numEntityBlocks	:	integer.
numNodes	:	integer.
minNodeTag	:	integer.
maxNodeTag	:	integer
EntityBlocks	:	array of EntityBlock structure.

Fields of EntityBlocks structure of Nodes

entityDim	:	integer.
entityTag	:	integer.
parametric	:	integer.
numNodes	:	integer.
nodeTags	:	1-by-numNodes array of integer.
Nodes	:	3-by-numNodes array of double.

Fields of Elements structure

numEntityBlocks	:	integer.
numElements	:	integer.
minElementTag	:	integer.
maxElementTag	:	integer
EntityBlocks	:	array of EntityBlock structure.
ElementTypes	:	array of .

Fields of EntityBlocks structure of Elements

entityDim	:	integer.
entityTag	:	integer.
elementType	:	integer.
elementDesc	:	structure returned by function gmsh.elm_type_desc(elementType).
numElementsBlock	:	integer.
nodeTags	:	n-by-numElementsBlock array. n depends of elementType: n = elementDesc.nb_nodes
elementTags	:	1-by-numElementsBlock array

Fields of PeriodicLink

entityDim	:	integer.
entityTag	:	integer.
entityTagMaster	:	integer.
numAffine	:	
values	:	.
numCorrespondingNodes	:	.
nodeTags	:	
nodeTagMasters	:	

4.3 Sample 1

The 2d .geo file *condenser.geo* is used to create a .msh file : `condenser-25.msh`. This `.msh` file contains only 1 (2-node line) and 2 (3-node triangle) *elm-type*.

Matlab code with output

```
meshfile=fc_oogmsh.gmsh.buildmesh('condenser',25,'verbose',0,'force',true);
Gh = fc_oogmsh.ooGmsh4(meshfile)
```

```
Gh =

  fc_oogmsh.ooGmsh4 with properties:
                  q: (2x49338 double)
                 nq: 49338 double
                dim: 2 double
                  d: 2 double
           toGlobal: (1x49338 double)
           meshfile: (1x110 char)
    partitionnedfile: 0 logical
          MeshFormat: (1x1 struct)
       PhysicalNames: []
            Entities: (1x1 struct)
  PartitionedEntities: []
              Nodes: (1x1 struct)
           Elements: (1x1 struct)
       PeriodicLinks: []
              Info: (1x1 struct)
            orders: 1 double
```

4.4 Sample 2

The 3d .geo file *cylinderkey.geo* is used to create a .msh file : `cylinderkey-10.msh`. This `.msh` file contains 1 (2-node line), 2 (3-node triangle) and 4 (4-node tetrahedron) *elm-type*.

Matlab code with output

```
meshfile=fc_oogmsh.gmsh.buildmesh3d('cylinderkey',10, ...
    'verbose',0,'force',true);
Gh = fc_oogmsh.ooGmsh4(meshfile)
```

```
ET =

    1    2    4


Gh =

  fc_oogmsh.ooGmsh4 with properties:
                   q: (3x5863 double)
                  nq: 5863 double
                 dim: 3 double
                   d: 3 double
            toGlobal: (1x5863 double)
            meshfile: (1x112 char)
      partitionnedfile: 0 logical
          MeshFormat: (1x1 struct)
        PhysicalNames: []
            Entities: (1x1 struct)
    PartitionedEntities: []
               Nodes: (1x1 struct)
            Elements: (1x1 struct)
        PeriodicLinks: []
                Info: (1x1 struct)
              orders: 1 double
```

4.5 Sample 3

The 3d .geo file *ball8.geo* is used to create a 3d surface .msh file : `ball8-50.msh`. This `.msh` file contains 1 (2-node line), 2 (3-node triangle) and 15 (1-node point) *elm-type*.

Matlab code with output

```
meshfile=fc_oogmsh.gmsh.buildmesh3ds('ball8',25, 'verbose',0,'force',true);
Gh = fc_oogmsh.ooGmsh4(meshfile)
```

```
ET =

    1    2   15


Gh =

  fc_oogmsh.ooGmsh4 with properties:
                   q: (3x9740 double)
                  nq: 9740 double
                 dim: 3 double
                   d: 2 double
            toGlobal: (1x9740 double)
            meshfile: (1x106 char)
      partitionnedfile: 0 logical
          MeshFormat: (1x1 struct)
        PhysicalNames: []
            Entities: (1x1 struct)
    PartitionedEntities: []
               Nodes: (1x1 struct)
            Elements: (1x1 struct)
        PeriodicLinks: []
                Info: (1x1 struct)
              orders: 1 double
```

5 ooGmsh2 class (version 2.2)

The ooGmsh2 class can be used to read `gmsh` mesh files with the MSH ASCII file format (version 2.2) described for example in [3], section 9.1. A MSH file can contain various mesh elements which are identified by an *elm-type* integer given in Appendix A. One can also refer to the fc_oogmsh.gmsh.elm_type_desc function, described in Appendix B.1, to obtain information on a given *elm-type*.

When reading a MSH file (format 2.2) generated by `gmsh`, we split the mesh elements by *elm-type* and generate an array of Elmt structure. The dimension of this array is the number of differents *elm-type* founds on the .msh file.

The `Elmt` structure is given by

Fields of Elmt structure

type	:	integer, refers to the type of the element : 1 for 2-node line, 2 for 3-node triangle, ... See the *elm-type* description of [3], section 9.1. Informations on a given **type** can be obtained by using elt=fc_oogmsh.gmsh.elm_type_desc(**type**).
geo	:	string, contains the kind of geometry: 'line', 'triangle', 'tetrahedron', ...
d	:	integer, space dimension or d-simplex.
order	:	integer, order of the element.
nme	:	integer, number of mesh elements.
me	:	array of nb_nodes-by-nme integers, connectivity array. nb_nodes is equal to elt.nb_nodes where elt=fc_oogmsh.gmsh.elm_type_desc(**type**).
phys_lab	:	array of nme-by-... integers, physical labels of the elements.
geo_lab	:	array of nme-by-... integers, geometrical labels of the elements.
nb_parts	:	array of nme-by-1 integers, number of mesh partitions to which the element belongs.
part_lab	:	array of nme-by-max(nb_parts) integers, part_lab$(i, 1 : \text{nb_parts}(i))$ contains all the partitions index to which the i-th element belongs.

The ooGmsh2 class was created to store a maximum of(all the) information(s) contained in the .msh file. The properties of this class are:

Properties of ooGmsh class

dim	:	integer space dimension
n_q	:	integer number of vertices/nodes
q	:	dim-by-n_q array of reals array of vertex coordinates
types	:	array of integers List of the element types found in the mesh file.
orders	:	array of integers List of the orders of the element types found in the mesh file.
sElts	:	array of `Elmt` structure One `Elmt` structure by element type, such that sElts(i) contains all the elements of type types(i) and order orders(i).

5.1 Methods

5.1.1 ooGmsh2 constructor

The ooGmsh2 class have only one constructor :

```
Gh=fc_oogmsh.ooGmsh2(meshfile)
```

where meshfile is the name of ... a mesh file

Matlab code with output

```
fprintf('1) Building the mesh\n')
meshfile=fc_oogmsh.gmsh.buildmesh2d('disk3holes',15, ...
    'verbose',0,'force',true,'MshFileVersion','2.2');
fprintf('2) Reading the mesh\n');
Gh = fc_oogmsh.ooGmsh2(meshfile);
fprintf('-> Gh is an ooGmsh2 object containing a MSH file version ...
    %s\n',Gh.MeshFormat.version)
fprintf('3) Displaying Gh\n');
Gh
```

```
1) Building the mesh
2) Reading the mesh
-> Gh is an ooGmsh2 object containing a MSH file version 2.2
3) Displaying Gh

Gh =

  fc_oogmsh.ooGmsh2 with properties:
                  q: (2x910 double)
                dim: 2 double
                 nq: 910 double
                  d: 2 double
             sElts: (2x1 struct)
           toGlobal: (1x910 double)
    partitionnedfile: 0 logical
             orders: 1 double
              types: (1x2 int32)
         MeshFormat: (1x1 struct)
           meshfile: (1x111 char)
               Info: (1x1 struct)
              debug: (1x1 struct)
```

5.1.2 `info` method

```
info(Gh)
Gh.info()
Gh.info(Key, Value, ...)
```

Description

Gh.**info**()

print informations on class fields with 3 levels of recursivity (i.e. field of field of field).

Gh.**info**(Key, Value, ...)

specifies function options using one or more Key,Value pair arguments. The Key options can be

- 'maxlevel' : level of recursivity, default is 3.
- 'tab' : number of space characters between two levels of recursivity, default is 4.

Matlab code with output

```
meshfile=fc_oogmsh.gmsh.buildmesh2d('disk3holes',15, ...
    'verbose',0,'force',true,'MshFileVersion','2.2');
Gh = fc_oogmsh.ooGmsh2(meshfile);
Gh.info('maxlevel',2);
```

```
fc_oogmsh.ooGmsh2 with properties:
  [1] q : [2 910] double
  [1] dim : [1 1] double
  [1] nq : [1 1] double
  [1] d : [1 1] double
  [1] sElts : [2 1] struct
      [2] type : [1 1] int32
      [2] geo : [1 4] char
      [2] d : [1 1] double
      [2] order : [1 1] double
      [2] me : [2 146] double
      [2] nme : [1 1] double
      [2] phys_lab : [146  1] double
      [2] geo_lab : [146  1] double
      [2] part_lab : [0 0] double
      [2] nb_parts : [146  1] double
  [1] toGlobal : [1 910] double
  [1] partitionnedfile : [1 1] logical
  [1] orders : [1 1] double
  [1] types : [1 2] int32
  [1] MeshFormat : [1 1] struct
      [2] version : [1 3] char
      [2] file_type : [1 1] double
      [2] data_size : [1 1] double
  [1] meshfile : [1 111] char
  [1] Info : [1 1] struct
      [2] meshfile : [1 111] char
  [1] debug : [1 1] struct
      [2] Mt : [1824  55] int32
```

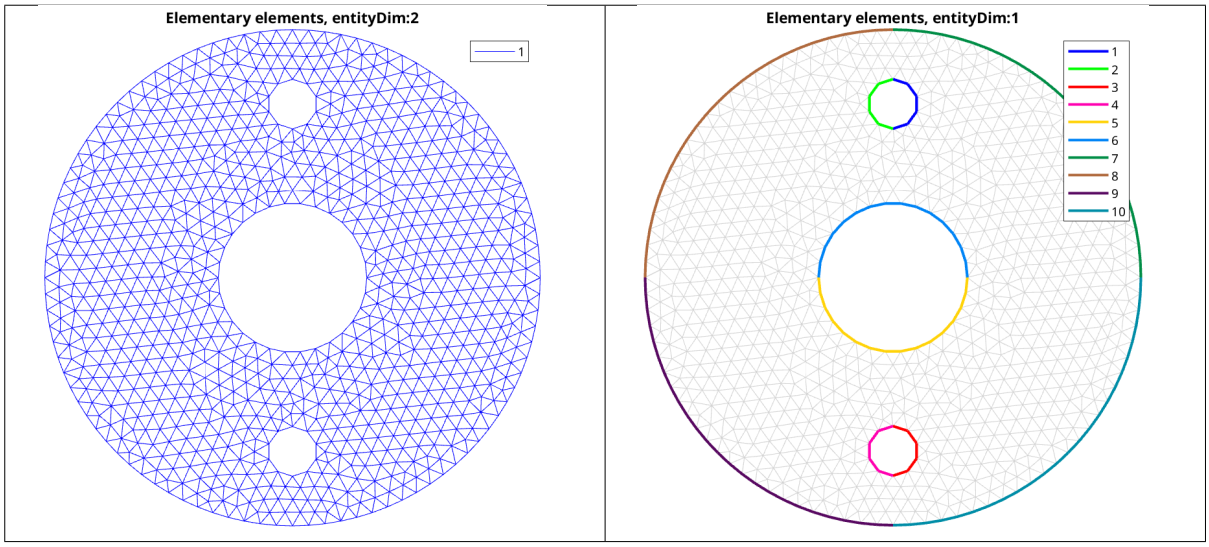


Figure 3: *Elementary Tag* elements of the *geofile* `disk3holes.geo`

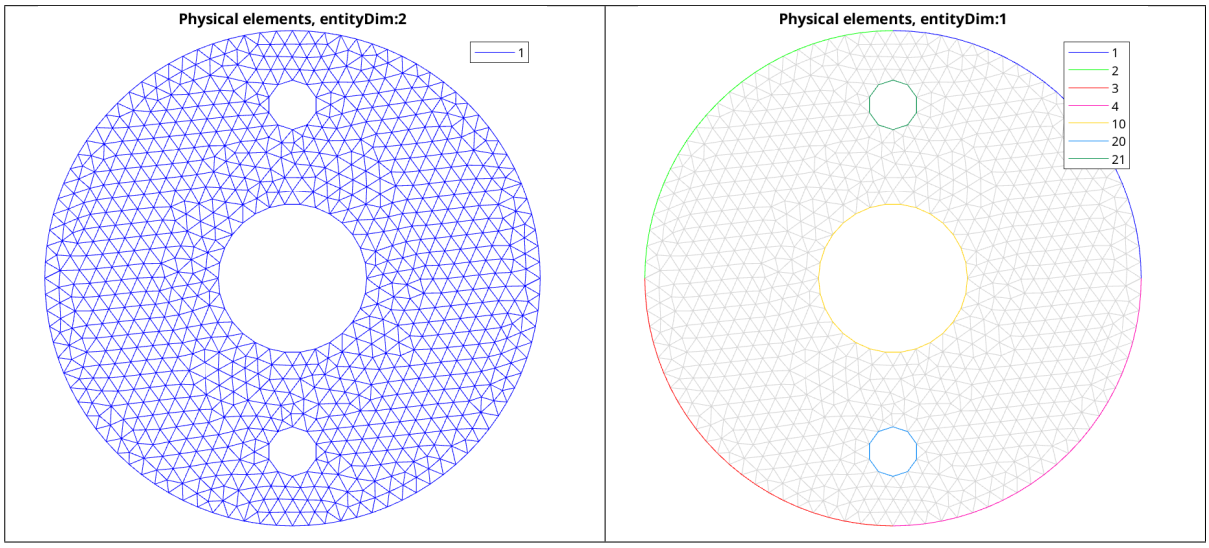


Figure 4: *Physical Tag* elements of the *geofile* `disk3holes.geo`

In the *geofile* `disk3holes.geo` the *Physical Tags* are created from the *Elementary Tags* as follow

```
...
Physical Line(10) = {6, 5};
Physical Line(21) = {2, 1};
Physical Line(20) = {4, 3};
Physical Line(1) = {7};
Physical Line(2) = {8};
Physical Line(3) = {9};
Physical Line(4) = {10};
Physical Surface(1) = {1};
```

5.1.3 get_ElementaryTags method

```
eltags=get_ElementaryTags(Gh,EltType)
eltags=Gh.get_ElementaryTags(EltType)
```

Description

`eltags=Gh.get_ElementaryTags(EltType)`

returns all the elementary tags associated with elements of type EltType as an array with unique elements. EltType is described in Section A. For example, EltType is 1 for `2-nodes line` (i.e 1-simplex of order 1), EltType is 2 for `3-nodes triangle` (i.e 2-simplex of order 1) and EltType is 4 for `4-nodes tetrahedron` (i.e 3-simplex of order 1).

Matlab code with output

```
eltags1=Gh.get_ElementaryTags(1)
eltags2=Gh.get_ElementaryTags(2)
```

```
eltags1 =

    1    2    3    4    5    6    7    8    9   10

eltags2 =

    1
```

5.1.4 get_PhysicalTags method

```
phtags=get_PhysicalTags(Gh,EltType)
phtags=Gh.get_PhysicalTags(EltType)
```

Description

phtags=Gh.get_PhysicalTags(EltType)

returns all the physical tags associated with elements of type EltType as an array with unique elements. EltType is described in Section A.

Matlab code with output

```
phtags1=Gh.get_PhysicalTags(1)
phtags2=Gh.get_PhysicalTags(2)
```

```
phtags1 =

    1    2    3    4   10   20   21

phtags2 =

    1
```

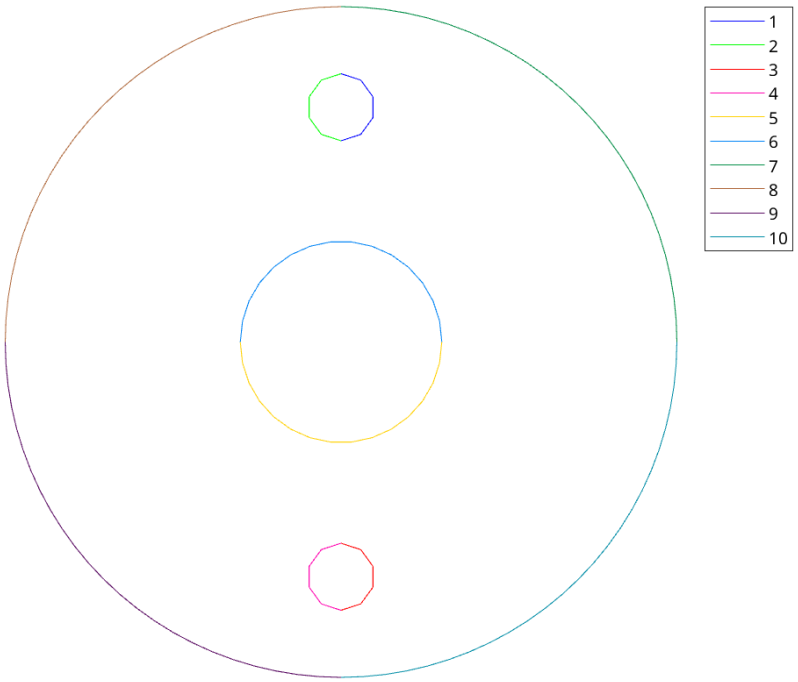
5.1.5 get_me_ElementaryTag method

```
me=get_me_ElementaryTag(Gh,EltType,EltTag)
me=Gh.get_me_ElementaryTag(EltType,EltTag)
```

Description

me=Gh.get_me_ElementaryTag(EltType,EltTag)

returns me the connectivity array of mesh elements of type and *elementary tag* given respectively by EltType and EltTag. This array is associated with the Gh.q nodes/vertices array.




```
meshfile=fc_oogmsh.gmsh.buildmesh2d('disk3holes',15, 'verbose',0,'force',true,'MshFileVersion','2.2');
Gh = fc_oogmsh.ooGmsh2(meshfile);
eltags1=Gh.get_ElementaryTags(1);
n1=length(eltags1);
colors = fc_tools.graphics.selectColors(length(eltags1));
figure(1)
hold on
for i=1:n1
  me=Gh.get_me_ElementaryTag(1,eltags1(i));
  h(i)=fc_graphics4mesh.plotmesh(Gh.q,me,'color',colors(i,:));
  clegend{i}=num2str(eltags1(i));
end
legend(h,clegend,'Location','NorthEastOutside','AutoUpdate','off');
axis image;axis off
```

Listing 5: Plot curves mesh elements by using get_me_ElementaryTags function and graphical function fc_graphics4mesh.plotmesh

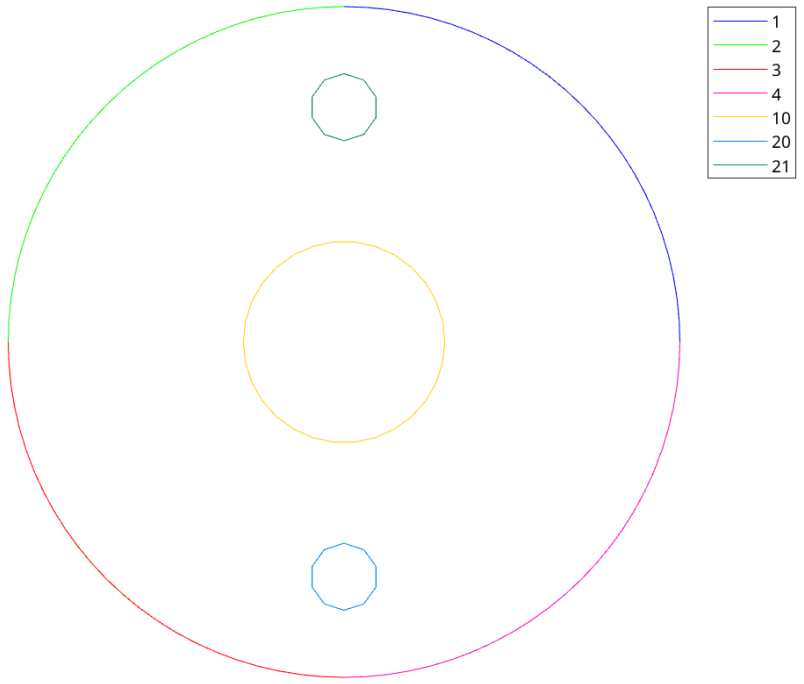
5.1.6 get_me_PhysicalTag method

```
me=get_me_PhysicalTag(Gh,EltType,PhysicalTag)
me=Gh.get_me_PhysicalTag(EltType,PhysicalTag)
```

Description

get_me_PhysicalTag(Gh,EltType,PhysicalTag)

returns me the connectivity array of mesh elements of type and *physical tag* given respectively by EltType and PhysicalTag.




```
meshfile=fc_oogmsh.gmsh.buildmesh2d('disk3holes',15, 'verbose',0,'force',true,'MshFileVersion','2.2');
Gh = fc_oogmsh.ooGmsh2(meshfile);
phtags1=Gh.get_PhysicalTags(1);
n1=length(phtags1);
colors = fc_tools.graphics.selectColors(length(phtags1));
figure(1)
hold on
for i=1:n1
  me=Gh.get_me_PhysicalTag(1,phtags1(i));
  h(i)=fc_graphics4mesh.plotmesh(Gh.q,me,'color',colors(i,:));
  clegend{i}=num2str(phtags1(i));
end
legend(h,clegend,'Location','NorthEastOutside','AutoUpdate','off');
axis image;axis off
```

Listing 6: Plot curves mesh elements by using get_me_PhysicalTags function and graphical function fc_graphics4mesh.plotmesh

5.1.7 get_localmesh_ElementaryTag method

```
[q,me]=Gh.get_localmesh_ElementaryTag(EltType,EltTag)
[q,me,toGlobal]=Gh.get_localmesh_ElementaryTag(EltType,EltTag)
```

[q,me]=Gh.get_localmesh_ElementaryTag(EltType,EltTag)

returns the *local* nodes/vertices array q and the *local* connectivity array me of the element of type EltType and with *elementary tag* given by EltTag.

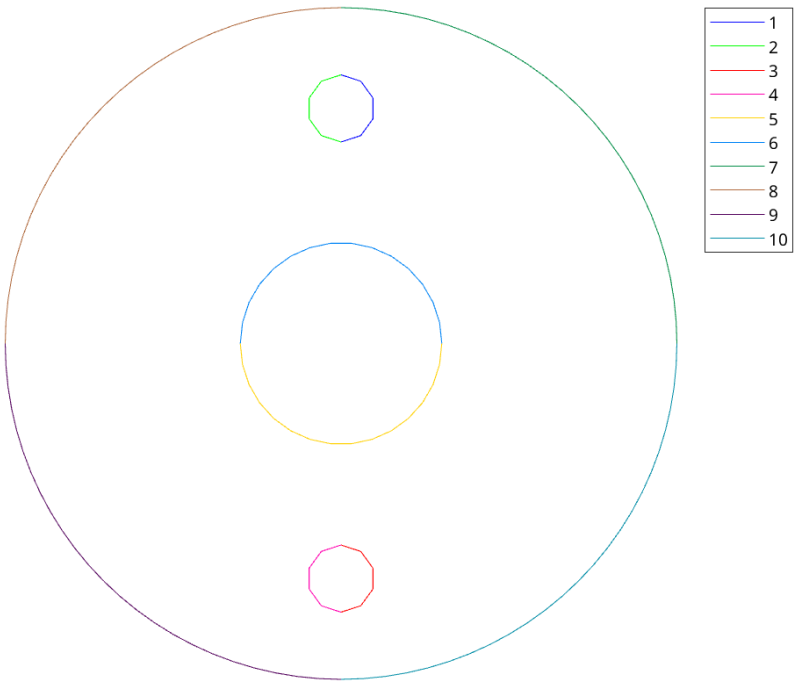
[q,me,toGlobal]=Gh.get_localmesh_ElementaryTag(EltType,EltTag)

Also returns the *global* tags array toGlobal such that Gh.q(:,toGlobal) is equal to q.

Matlab code with output

```
Gh =fc_oogmsh.ooGmsh2(meshfile);
eltags1=Gh.get_ElementaryTags(1);
n1=length(eltags1);
for i=1:n1
  [q,me,toG]=Gh.get_localmesh_ElementaryTag(1,eltags1(i));
  fprintf('d=1,␣Elem.Tag=%d,␣...
      error=%.5e\n',eltags1(i),norm(Gh.q(:,toG)-q,Inf))
end
```

```
d=1, Elem.Tag=1, error=0.00000e+00
d=1, Elem.Tag=2, error=0.00000e+00
d=1, Elem.Tag=3, error=0.00000e+00
d=1, Elem.Tag=4, error=0.00000e+00
d=1, Elem.Tag=5, error=0.00000e+00
d=1, Elem.Tag=6, error=0.00000e+00
d=1, Elem.Tag=7, error=0.00000e+00
d=1, Elem.Tag=8, error=0.00000e+00
d=1, Elem.Tag=9, error=0.00000e+00
d=1, Elem.Tag=10, error=0.00000e+00
```




```
meshfile=fc_oogmsh.gmsh.buildmesh2d('disk3holes',15, 'verbose',0,'force',true,'MshFileVersion','2.2');
Gh = fc_oogmsh.ooGmsh2(meshfile);
eltags1=Gh.get_ElementaryTags(1);
n1=length(eltags1);
colors = fc_tools.graphics.selectColors(length(eltags1));
figure(1)
hold on
 for i=1:n1
  [q,me]=Gh.get_localmesh_ElementaryTag(1,eltags1(i));
  h(i)=fc_graphics4mesh.plotmesh(q,me,'color',colors(i,:));
  clegend{i}=num2str(eltags1(i));
 end
legend(h,clegend,'Location','NorthEastOutside','AutoUpdate','off');
axis image;axis off
```

Listing 7: Plot curves mesh elements by using get_localmesh_ElementaryTag function and graphical function fc_graphics4mesh.plotmesh

5.1.8 get_localmesh_PhysicalTag method

```
[q,me]=Gh.get_localmesh_PhysicalTag(EltType,PhysicalTag)
[q,me,toGlobal]=Gh.get_localmesh_PhysicalTag(EltType,PhysicalTag)
```

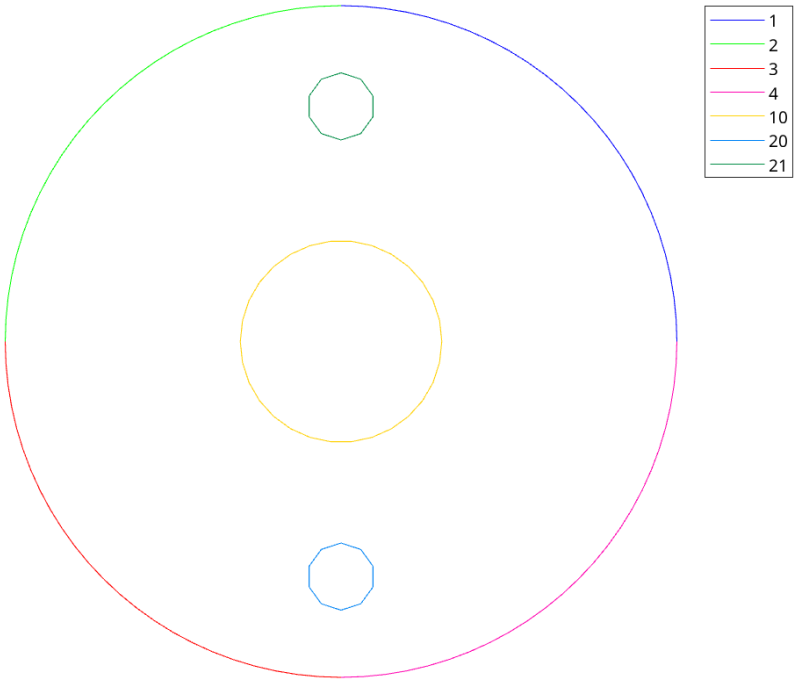
[q,me]=Gh.get_localmesh_PhysicalTag(EltType,PhysicalTag)

returns the *local* nodes/vertices array q and the *local* connectivity array me of the elements of type

EltType and with *PhysicalTag* given by PhysicalTag.

```
[q,me,toGlobal]=Gh.get_localmesh_PhysicalTag(EltType,PhysicalTag)
```

Also returns the *global* tags array toGlobal such that Gh.q(:,toGlobal) is equal to q.




```
meshfile=fc_oogmsh.gmsh.buildmesh2d('disk3holes',15, 'verbose',0,'force',true,'MshFileVersion','2.2');
Gh = fc_oogmsh.ooGmsh2(meshfile);
phtags1=Gh.get_PhysicalTags(1);
n1=length(phtags1);
colors = fc_tools.graphics.selectColors(length(phtags1));
figure(1)
hold on
for i=1:n1
  [q,me]=Gh.get_localmesh_PhysicalTag(1,phtags1(i));
  h(i)=fc_graphics4mesh.plotmesh(q,me,'color',colors(i,:));
  clegend{i}=num2str(phtags1(i));
end
legend(h,clegend,'Location','NorthEastOutside','AutoUpdate','off');
axis image;axis off
```

Listing 8: Plot curves mesh elements by using get_localmesh_PhysicalTag function and graphical function fc_graphics4mesh.plotmesh

5.2 Sample 1

The 2d .geo file *condenser.geo* is used to create a .msh file : `condenser-25.msh`. This `.msh` file contains only 1 (2-node line) and 2 (3-node triangle) *elm-type*.

Matlab code with output

```
meshfile=fc_oogmsh.gmsh.buildmesh('condenser',25,'verbose',0, ...
    'force',true,'MshFileVersion','2.2');
Gh = fc_oogmsh.ooGmsh2(meshfile)
```

```
Gh =

  fc_oogmsh.ooGmsh2 with properties:
                  q: (2x49338 double)
                dim: 2 double
                 nq: 49338 double
                  d: 2 double
               sElts: (3x1 struct)
          toGlobal: (1x49338 double)
    partitionnedfile: 0 logical
             orders: 1 double
              types: (1x3 int32)
         MeshFormat: (1x1 struct)
           meshfile: (1x110 char)
               Info: (1x1 struct)
              debug: (1x1 struct)
```

5.3 Sample 2

The 3d .geo file *cylinderkey.geo* is used to create a .msh file : `cylinderkey-10.msh`. This `.msh` file contains 1 (2-node line), 2 (3-node triangle) and 4 (4-node tetrahedron) *elm-type*.

Matlab code with output

```
meshfile=fc_oogmsh.gmsh.buildmesh3d('cylinderkey',10,'verbose',0, ...
    'force',true,'MshFileVersion','2.2');
Gh = fc_oogmsh.ooGmsh2(meshfile)
```

```
Gh =

  fc_oogmsh.ooGmsh2 with properties:
                 q: (3x5863 double)
               dim: 3 double
                nq: 5863 double
                 d: 3 double
             sElts: (3x1 struct)
          toGlobal: (1x5863 double)
    partitionnedfile: 0 logical
            orders: 1 double
             types: (1x3 int32)
        MeshFormat: (1x1 struct)
          meshfile: (1x112 char)
              Info: (1x1 struct)
             debug: (1x1 struct)
```

5.4 Sample 3

The 3d .geo file *ball8.geo* is used to create a 3d surface .msh file : `ball8-50.msh`. This `.msh` file contains 1 (2-node line), 2 (3-node triangle) and 15 (1-node point) *elm-type*.

Matlab code with output

```
meshfile=fc_oogmsh.gmsh.buildmesh3ds('ball8',50,'verbose',0, ...
    'force',true,'MshFileVersion','2.2');
Gh = fc_oogmsh.ooGmsh2(meshfile)
```

```
Gh =

  fc_oogmsh.ooGmsh2 with properties:
                 q: (3x37799 double)
               dim: 3 double
                nq: 37799 double
                 d: 2 double
             sElts: (3x1 struct)
          toGlobal: (1x37799 double)
    partitionnedfile: 0 logical
            orders: 1 double
             types: (1x3 int32)
        MeshFormat: (1x1 struct)
          meshfile: (1x106 char)
              Info: (1x1 struct)
             debug: (1x1 struct)
```

A Element type

In a .msh file the kind of mesh elements are identified by their *elm-type* integer values :

Table 1:

elm-type	**description**
1	2-node line
2	3-node triangle
3	4-node quadrangle
4	4-node tetrahedron

Continued on next page

Table 1: (Continued)

5	8-node hexahedron
6	6-node prism
7	5-node pyramid
8	3-node second order line (2 nodes associated with the vertices and 1 with the edge)
9	6-node second order triangle (3 nodes associated with the vertices and 3 with the edges)
10	9-node second order quadrangle (4 nodes associated with the vertices, 4 with the edges and 1 with the face)
11	10-node second order tetrahedron (4 nodes associated with the vertices and 6 with the edges)
12	27-node second order hexahedron (8 nodes associated with the vertices, 12 with the edges, 6 with the faces and 1 with the volume)
13	18-node second order prism (6 nodes associated with the vertices, 9 with the edges and 3 with the quadrangular faces)
14	14-node second order pyramid (5 nodes associated with the vertices, 8 with the edges and 1 with the quadrangular face)
15	1-node point
16	8-node second order quadrangle (4 nodes associated with the vertices and 4 with the edges)
17	20-node second order hexahedron (8 nodes associated with the vertices and 12 with the edges)
18	15-node second order prism (6 nodes associated with the vertices and 9 with the edges)
19	13-node second order pyramid (5 nodes associated with the vertices and 8 with the edges)
20	9-node third order incomplete triangle (3 nodes associated with the vertices, 6 with the edges)
21	10-node third order triangle (3 nodes associated with the vertices, 6 with the edges, 1 with the face)
22	12-node fourth order incomplete triangle (3 nodes associated with the vertices, 9 with the edges)
23	15-node fourth order triangle (3 nodes associated with the vertices, 9 with the edges, 3 with the face)
24	15-node fifth order incomplete triangle (3 nodes associated with the vertices, 12 with the edges)
25	21-node fifth order complete triangle (3 nodes associated with the vertices, 12 with the edges, 6 with the face)
26	4-node third order edge (2 nodes associated with the vertices, 2 internal to the edge)
27	5-node fourth order edge (2 nodes associated with the vertices, 3 internal to the edge)
28	6-node fifth order edge (2 nodes associated with the vertices, 4 internal to the edge)
29	20-node third order tetrahedron (4 nodes associated with the vertices, 12 with the edges, 4 with the faces)
30	35-node fourth order tetrahedron (4 nodes associated with the vertices, 18 with the edges, 12 with the faces, 1 in the volume)

Continued on next page

Table 1: (Continued)

31	56-node fifth order tetrahedron (4 nodes associated with the vertices, 24 with the edges, 24 with the faces, 4 in the volume)
92	64-node third order hexahedron (8 nodes associated with the vertices, 24 with the edges, 24 with the faces, 8 in the volume)
93	125-node fourth order hexahedron (8 nodes associated with the vertices, 36 with the edges, 54 with the faces, 27 in the volume)

B Other functions

B.1 function fc_oogmsh.gmsh.elm_type_desc

This function returns a structure which contains some informations on a `gmsh` *elt-type* described in Appendix A.

Syntaxe

```
elt=fc_oogmsh.gmsh.elm_type_desc(type)
```

Matlab code with output

```
elt2=fc_oogmsh.gmsh.elm_type_desc(2)
elt4=fc_oogmsh.gmsh.elm_type_desc(4)
elt11=fc_oogmsh.gmsh.elm_type_desc(11)
```

```
elt2 =

  struct with fields:

       define: 'MSH_TRI_3'
     elm_type: 2
         desc: '3-node triangle'
     nb_nodes: 3
        order: 1
   incomplete: 0
            d: 2
          geo: 'triangle'


elt4 =

  struct with fields:

       define: 'MSH_TET_4'
     elm_type: 4
         desc: '4-node tetrahedron'
     nb_nodes: 4
        order: 1
   incomplete: 0
            d: 3
          geo: 'tetrahedron'


elt11 =

  struct with fields:

       define: 'MSH_TET_10'
     elm_type: 11
         desc: '10-node second order tetrahedron (4 nodes associated with the vertices and 6 with the edges)'
     nb_nodes: 10
        order: 2
   incomplete: 0
            d: 3
          geo: 'tetrahedron'
```

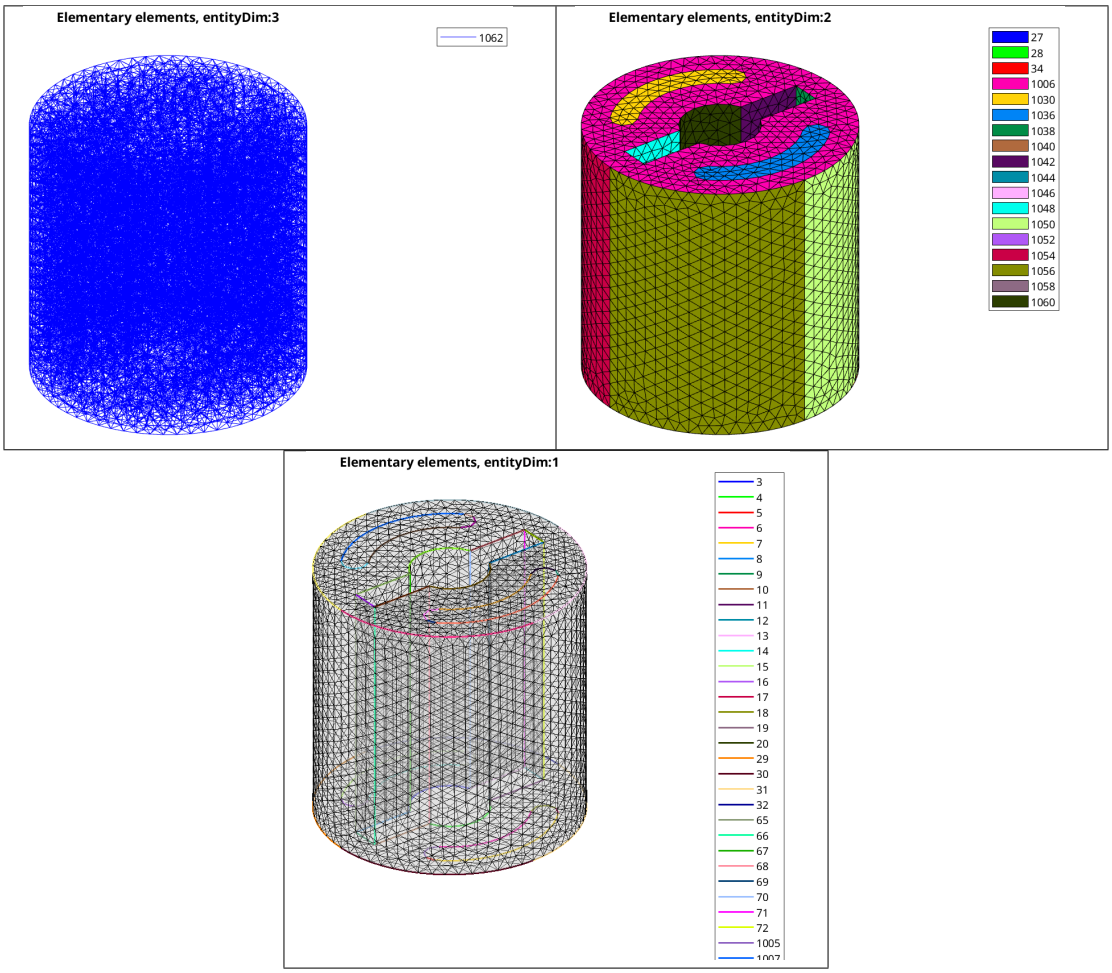
B.2 method plotElementaryElements

This function plot *Elementary Elements* of an ooGmsh2 or ooGmsh4 object of *Element Type*

- 1, *2-node line* elements,
- 2, *3-node triangle* elements,
- 4, *4-node tetrahedron* elements.

This function uses the `fc-graphics4mesh` toolbox [1] version 0.1.7.

```
Gh.plotElementaryElements()
plotElementaryElements(Gh)
```




```
meshfile=fc_oogmsh.gmsh.buildmesh3d('cylinderkey',10, 'verbose',0,'force',true);
Gh = fc_oogmsh.ooGmsh4(meshfile);
Gh.plotElementaryElements()
```

Listing 9: Using plotElementaryElements method

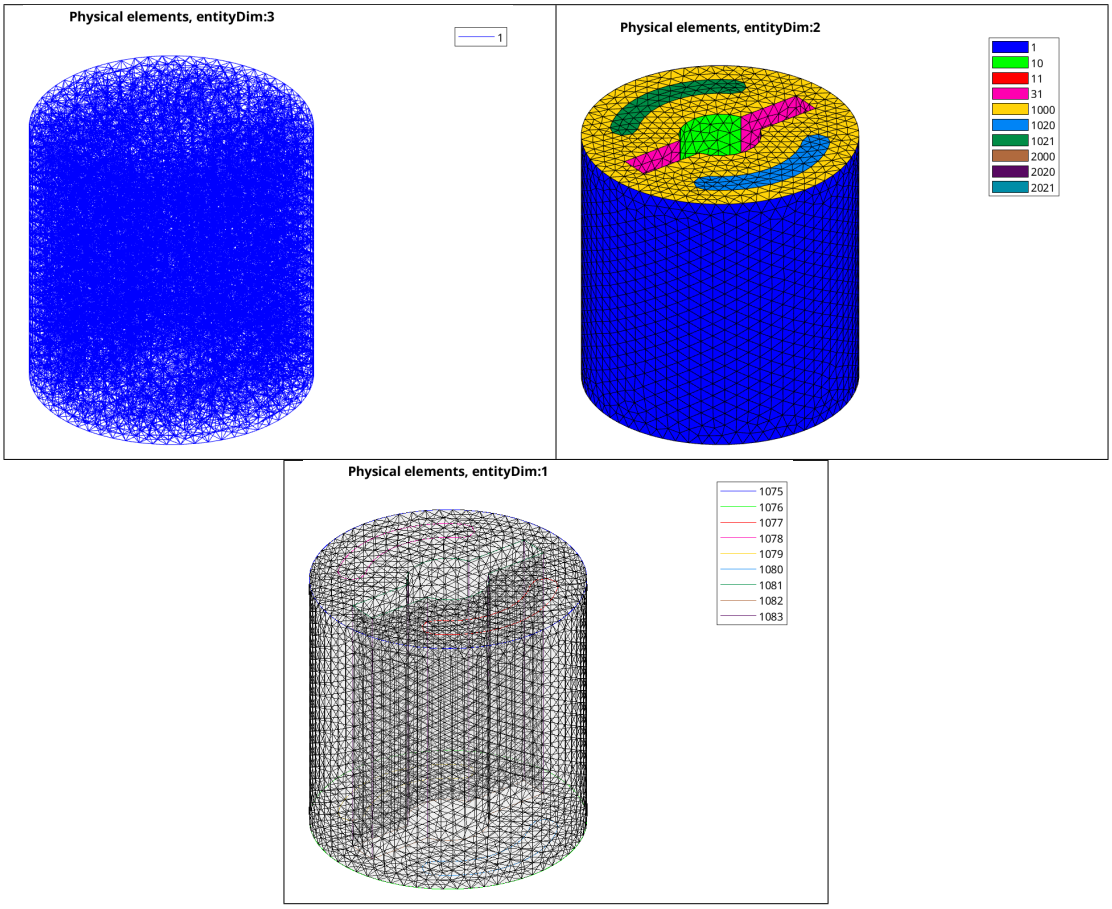
B.3 method `plotPhysicalElements`

This function plot *Physical Tags* of an ooGmsh2 or ooGmsh4 object of *Element Type*

- 1, *2-node line* elements,
- 2, *3-node triangle* elements,
- 4, *4-node tetrahedron* elements.

This function uses the `fc-graphics4mesh` toolbox [1] version 0.1.7.

```
Gh.plotPhysicalElements()
plotPhysicalElements(Gh)
```




```
meshfile=fc_oogmsh.gmsh.buildmesh3d('cylinderkey',10, 'verbose',0,'force',true);
Gh = fc_oogmsh.ooGmsh4(meshfile);
plotPhysicalElements(Gh)
```

Listing 10: Using fc_oogmsh.plotPhysicalElements function

B.4 method plotPartitionElements

This function can be used with partitioned mesh file built with one of the following functions:

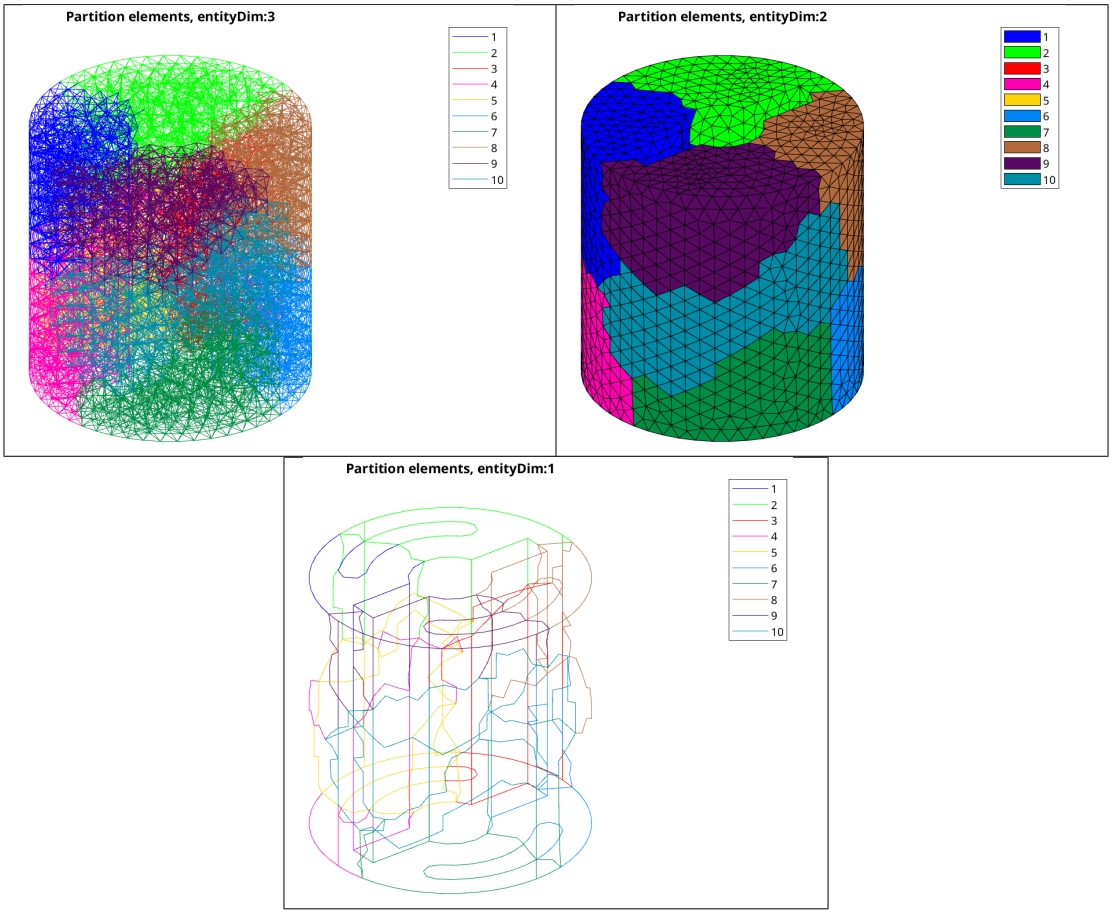
fc_oogmsh.gmsh.buildpartmesh3d,
fc_oogmsh.gmsh.buildpartmesh3ds,
fc_oogmsh.gmsh.buildpartmesh2d.

This function plot *Partition Tags* of an ooGmsh2 or ooGmsh4 object of *Element Type*

- 1, *2-node line* elements,
- 2, *3-node triangle* elements,
- 4, *4-node tetrahedron* elements.

This function uses the `fc-graphics4mesh` toolbox [1] version 0.1.7.

```
Gh.plotPartitionElements()
plotPartitionElements(Gh)
```




```
meshfile=fc_oogmsh.gmsh.buildpartmesh3d('cylinderkey',10,'N',8,'verbose',0,'force',true);
Gh = fc_oogmsh.ooGmsh4(meshfile);
plotPartitionElements(Gh)
```

Listing 11: Using fc_oogmsh.plotPartitionElements function

B.5 method plotInterfaceElements

This function can be used with partitioned mesh file built with one of the following functions:

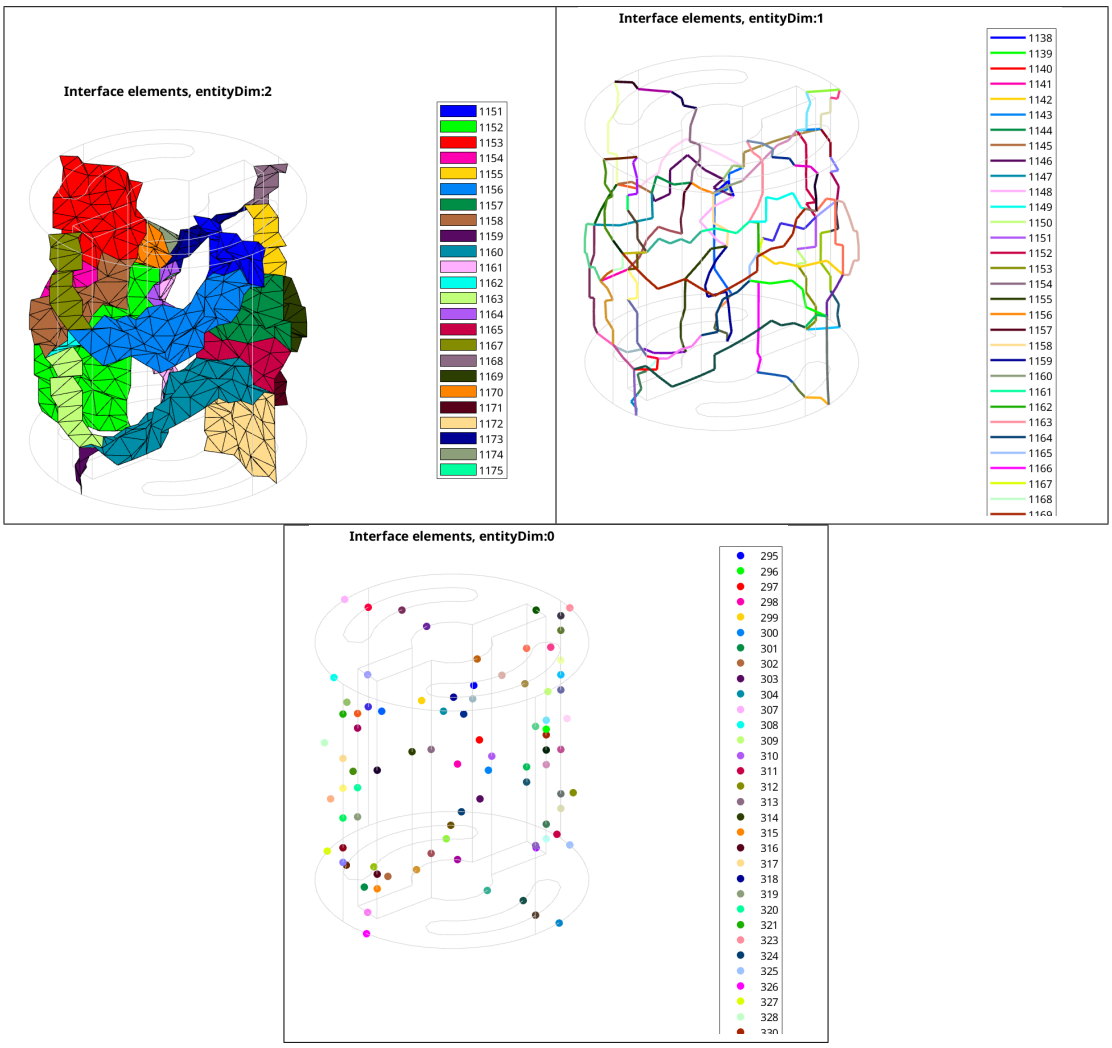
fc_oogmsh.gmsh.buildpartmesh3d,
fc_oogmsh.gmsh.buildpartmesh3ds,
fc_oogmsh.gmsh.buildpartmesh2d.

This function plot *Interface Tags* of the interfaces between partitions of an ooGmsh2 or ooGmsh4 object of *Element Type*

- 1, *2-node line* elements,
- 2, *3-node triangle* elements,
- 4, *4-node tetrahedron* elements.

This function uses the `fc-graphics4mesh` toolbox [1] version 0.1.7.

```
Gh.plotInterfaceElements()
plotInterfaceElements(Gh)
```




```
meshfile=fc_oogmsh.gmsh.buildpartmesh3d('cylinderkey',10,'N',8,'verbose',0,'force',true);
Gh = fc_oogmsh.ooGmsh4(meshfile); %
plotInterfaceElements(Gh)
```

Listing 12: Using fc_oogmsh.plotInterfaceElements function

B References

[1] F. Cuvelier. fc_graphics4mesh: a Matlab toolbox for displaying simplices meshes or datas on simplices meshes. `http://www.math.univ-paris13.fr/~cuvelier/software/`, 2017. User's Guide.

[2] C. Geuzaine and J.-F. Remacle. Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009.

[3] C. Geuzaine and J.-F. Remacle. Gmsh 2.15.0. `http://gmsh.info`, 2016.

[4] C. Geuzaine and J.-F. Remacle. Gmsh 4.2.1. `http://gmsh.info`, 2019.

Informations for git maintainers of the fc oogmsh Matlab toolbox

git informations on the toolboxes used to build this manual

```
-------------------------------------------------
   name : fc-oogmsh
    tag : 0.3.1
 commit : e52089e4a08f7084989a535be96dae219b899834
   date : 2025-01-28
   time : 06-40-27
 status : 0
-------------------------------------------------
   name : fc-tools
    tag : 0.0.36
 commit : 00c110c58dff7e001ec8130802d1725abf991f33
   date : 2025-01-26
   time : 05-09-25
 status : 0
-------------------------------------------------
   name : fc-bench
    tag : 0.1.4
 commit : b00bc133994a648d8909c4952f01659f8dbb5a8f
   date : 2025-01-26
   time : 05-13-31
 status : 0
-------------------------------------------------
   name : fc-amat
    tag : 0.1.4
 commit : c8ac405135a3606b8c2809d9d2dd71ee0989b5b8
   date : 2025-01-26
   time : 05-19-40
 status : 0
-------------------------------------------------
   name : fc-meshtools
    tag : 0.1.5
 commit : b7ff9340a05d6fb443c84cc27ca6b13339c2fd81
   date : 2025-01-26
   time : 07-33-01
 status : 0
-------------------------------------------------
   name : fc-graphics4mesh
    tag : 0.1.7
 commit : 795429326c1c20a60c186f27df2fefd24680a431
   date : 2025-01-28
   time : 08-12-35
 status : 0
-------------------------------------------------
```

git informations on the LaTeX package used to build this manual

```
-------------------------------------------------
   name : fctools
    tag :
 commit : 03d38737a795cdbf4e1a8754470e963cdfe83316
   date : 2025-01-24
   time : 09:58:52
 status : 1
-------------------------------------------------
```

Using the remote configuration repository:

url	ssh://lagagit/MCS/Cuvelier/Matlab/fc-config
commit	493bcec168b24d48a089286327db7bdd6bc6c55d