oogmsh Octave package, User's Guide*

version 0.1.0

François Cuvelier†

March 24, 2019

Abstract

This Octave package 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`) and to store its contains in more user-friendly form. This package must be regarded as a very simple interface between gmsh files and Octave. 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 0.1.0.a, compiled with Octave 5.1.0, and packages `fc-oogmsh`[0.1.0], `fc-tools`[0.0.27], `fc-meshtools`[0.1.0], `fc-graphics4mesh`[0.0.4], and using `gmsh` 4.2.2

†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 fc oogmsh Octave package is closely related to gmsh, see [4] or [1], 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 package was initialy created to make it possible from Octave to rapidly

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

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

This package was tested on various OS with gmsh (versions 4.2.2, 4.1.5, 4.0.7 and 3.0.6) and 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.

Firstly, we explain how to configure the fc oogmsh package for using `gmsh`. Thereafter, we describe the fc 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

ofc_oogmsh_install.m

or get it on the dedicated web page. Thereafter, one run it under Octave. This command download, extract and configure the *fc-oogmsh* package and the required *fc-tools* package 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 10

It can be directly given by using the `'gmsh_bin'` option of the install command:

```
>> ofc_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 Octave command

```
>> fc_oogmsh.configure('gmsh_bin','~/gmsh-4.2.2/bin/gmsh')
```

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

```
>> cd ~/Octave/packages
>> ofc_oogmsh_install
```

There is the output of the `ofc_oogmsh_install` command:

```
Parts of the <fc-oogmsh> Octave package.
Copyright (C) 2017-2019 F. Cuvelier

1- Downloading and extracting the packages
2- Setting the <fc-oogmsh> package
warning: isdir is obsolete; use isfolder or dir_in_loadpath instead
Write in ~/Octave/packages/fc-oogmsh-full/fc_oogmsh-0.1.0/configure_loc.m ...
3- Using packages :
   ->             fc-tools : 0.0.27
   ->         fc-meshtools : 0.1.0
   ->     fc-graphics4mesh : 0.0.4
 with            fc-oogmsh : 0.1.0
*** Using instructions
   To use the <fc-oogmsh> package:
   addpath('~/Octave/packages/fc-oogmsh-full/fc_oogmsh-0.1.0')
   fc_oogmsh.init()

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

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

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

and, for each Octave session, one have to set the package by:

```
>> addpath('~/Octave/packages/fc-oogmsh-full/fc-oogmsh-0.1.0')
>> 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 ~/Octave/packages/fc-oogmsh-full/fc_tools-0.0.27/configure_loc.m ...
Try to use default parameters!
 Use fc_meshtools.configure to configure.
Write in ~/Octave/packages/fc-oogmsh-full/fc_meshtools-0.1.0/configure_loc.m ...
Try to use default parameters!
 Use fc_graphics4mesh.configure to configure.
Write in ~/Octave/packages/fc-oogmsh-full/fc_graphics4mesh-0.0.4/configure_loc.m ...
Using fc_oogmsh[0.1.0] with fc_tools[0.0.27], fc_meshtools[0.1.0], fc_graphics4mesh[0.0.4].
Configured to use gmsh 4.0.1 with default MSH file format version 4
```

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

```
Using fc_oogmsh[0.1.0] with fc_tools[0.0.27], fc_meshtools[0.1.0], fc_graphics4mesh[0.0.4].
Configured to use gmsh 4.0.1 with default MSH file format version 4
```

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

`~/Octave/packages/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;'}`

- `'MshFileVersion'` : to specify the MSH file format version. `Value` could be
  - `'2.2'` if gmsh version $\geqslant$ 3.0.0,
  - `'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 `condenser11.geo` which is in the directory `geodir` of the package.

Octave code with output

```
disp('**** fc_oogmsh.gmsh.buildmesh2d : 1st call')
meshfile=fc_oogmsh.gmsh.buildmesh2d('condenser11',25,'force',true);
disp('**** fc_oogmsh.gmsh.buildmesh2d : 2nd call')
meshfile=fc_oogmsh.gmsh.buildmesh2d('condenser11',25);
```

```
**** fc_oogmsh.gmsh.buildmesh2d : 1st call
[fc-oogmsh] Input file : <fc-oogmsh>/geodir/2d/condenser11.geo
[fc-oogmsh] Overwritting mesh file <fc-oogmsh>/meshes/condenser11-25.msh
[fc-oogmsh] Starting building mesh <fc-oogmsh>/meshes/condenser11-25.msh with gmsh 4.2.2
[fc-oogmsh] Using command : gmsh -2 -setnumber N 25 -string "Mesh.MshFileVersion=4.1;" <fc-oogmsh>/geodir/2d/condenser11.geo -o ...
       <fc-oogmsh>/meshes/condenser11-25.msh
 Be patient...
[fc-oogmsh] Using gmsh 4.2.2 to write MSH file format version 4.1 in <fc-oogmsh>/meshes/condenser11-25.msh
**** fc_oogmsh.gmsh.buildmesh2d : 2nd call
```

Octave code with output

```
meshfile=fc_oogmsh.gmsh.buildmesh2d('condenser11',25,'force',true, ...
    'verbose',4, 'strings',{'Mesh.Algorithm=1;', 'Mesh.ScalingFactor=2;'});
```

```
[fc-oogmsh] Input file : <fc-oogmsh>/geodir/2d/condenser11.geo
[fc-oogmsh] Overwritting mesh file <fc-oogmsh>/meshes/condenser11-25.msh
[fc-oogmsh] Starting building mesh <fc-oogmsh>/meshes/condenser11-25.msh with gmsh 4.2.2
[fc-oogmsh] Using command : gmsh -2 -setnumber N 25 -string "Mesh.Algorithm=1;Mesh.ScalingFactor=2;Mesh.MshFileVersion=4.1;" ...
       <fc-oogmsh>/geodir/2d/condenser11.geo -o <fc-oogmsh>/meshes/condenser11-25.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/condenser11.geo -o <fc-oogmsh>/meshes/condenser11-25.msh' [Gmsh 4.2.2, 1 node, max. 1 thread]
Info    : Started on Sun Mar 24 07:58:00 2019
Info    : Reading '<fc-oogmsh>/geodir/2d/condenser11.geo'...
Info    : Reading '<fc-oogmsh>/geodir/2d/options01_data.geo'...
Info    : Done reading '<fc-oogmsh>/geodir/2d/options01_data.geo'
Info    : Reading '<fc-oogmsh>/geodir/2d/shape_functions.geo'...
Info    : Done reading '<fc-oogmsh>/geodir/2d/shape_functions.geo'
Info    : Removing duplicate mesh vertices...
Info    : Found 0 duplicate vertices
Info    : No duplicate vertices found
Info    : Done reading '<fc-oogmsh>/geodir/2d/condenser11.geo'
Info    : Meshing 1D...
Info    : Meshing curve 101 (Line)
Info    : Meshing curve 102 (Line)
Info    : Meshing curve 103 (Line)
Info    : Meshing curve 104 (Line)
Info    : Meshing curve 106 (Circle)
Info    : Meshing curve 107 (Circle)
Info    : Meshing curve 108 (Circle)
Info    : Meshing curve 109 (Circle)
Info    : Meshing curve 111 (Circle)
Info    : Meshing curve 112 (Circle)
Info    : Meshing curve 113 (Circle)
Info    : Meshing curve 114 (Circle)
Info    : Meshing curve 116 (Circle)
Info    : Meshing curve 117 (Circle)
Info    : Meshing curve 118 (Circle)
Info    : Meshing curve 119 (Circle)
Info    : Meshing curve 121 (Circle)
Info    : Meshing curve 122 (Circle)
Info    : Meshing curve 123 (Circle)
Info    : Meshing curve 124 (Circle)
Info    : Meshing curve 126 (Circle)
Info    : Meshing curve 127 (Circle)
Info    : Meshing curve 128 (Circle)
Info    : Meshing curve 129 (Circle)
Info    : Meshing curve 131 (Circle)
Info    : Meshing curve 132 (Circle)
Info    : Meshing curve 133 (Circle)
Info    : Meshing curve 134 (Circle)
Info    : Meshing curve 136 (Circle)
Info    : Meshing curve 137 (Circle)
Info    : Meshing curve 138 (Circle)
Info    : Meshing curve 139 (Circle)
Info    : Meshing curve 141 (Circle)
Info    : Meshing curve 142 (Circle)
Info    : Meshing curve 143 (Circle)
Info    : Meshing curve 144 (Circle)
Info    : Meshing curve 146 (Circle)
Info    : Meshing curve 147 (Circle)
Info    : Meshing curve 148 (Circle)
Info    : Meshing curve 149 (Circle)
Info    : Done meshing 1D (0.014957 s)
Info    : Meshing 2D...
Info    : Meshing surface 105 (Plane, MeshAdapt)
Info    : Meshing surface 110 (Plane, MeshAdapt)
Info    : Meshing surface 120 (Plane, MeshAdapt)
Info    : Meshing surface 130 (Plane, MeshAdapt)
Info    : Meshing surface 140 (Plane, MeshAdapt)
Info    : Meshing surface 150 (Plane, MeshAdapt)
Info    : Done meshing 2D (0.209784 s)
Info    : 3089 vertices 6311 elements
Info    : Writing '<fc-oogmsh>/meshes/condenser11-25.msh'...
Info    : Done writing '<fc-oogmsh>/meshes/condenser11-25.msh'
Info    : Stopped on Sun Mar 24 07:58:01 2019

[fc-oogmsh] Using gmsh 4.2.2 to write MSH file format version 4.1 in <fc-oogmsh>/meshes/condenser11-25.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.

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.

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 $\geqslant$ 3.0.0,
  - `'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 **meshfile** as output of the command :
`meshfile=gmsh.buildmesh2d('condenser11',25);`

Octave code with output

```
meshfile=fc_oogmsh.gmsh.buildmesh2d('condenser11',25,'verbose',0);
pmfile=fc_oogmsh.gmsh.buildpartmesh2d(meshfile,5,'force',true);
```

```
[fc-oogmsh] Input file : <fc-oogmsh>/meshes/condenser11-25.msh
[fc-oogmsh] Overwritting mesh file <fc-oogmsh>/meshes/condenser11-25-part5.msh
[fc-oogmsh] Starting building mesh <fc-oogmsh>/meshes/condenser11-25-part5.msh with gmsh 4.2.2
[fc-oogmsh] Using command : gmsh -2 -saveall -part 5 -string "Mesh.MshFileVersion=4.1;" <fc-oogmsh>/meshes/condenser11-25.msh -o ...
       <fc-oogmsh>/meshes/condenser11-25-part5.msh
 Be patient...
[fc-oogmsh] Using gmsh 4.2.2 to write MSH file format version 4.1 in <fc-oogmsh>/meshes/condenser11-25-part5.msh
```

Octave code with output

```
meshfile=fc_oogmsh.gmsh.buildmesh2d('condenser11',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/condenser11-25.msh
[fc-oogmsh] Overwritting mesh file <fc-oogmsh>/meshes/condenser11-25-part5.msh
[fc-oogmsh] Starting building mesh <fc-oogmsh>/meshes/condenser11-25-part5.msh with gmsh 4.2.2
[fc-oogmsh] Using command : gmsh -2 -saveall -part 5 -string "Mesh.MetisAlgorithm=3;Mesh.MshFileVersion=4.1;" <fc-oogmsh>/meshes/condenser11-25.msh -o ...
       <fc-oogmsh>/meshes/condenser11-25-part5.msh
 Be patient...
[fc-oogmsh] gmsh output :
Info    : Running '/home/cuvelier/bin/gmsh -2 -saveall -part 5 -string Mesh.MetisAlgorithm=3;Mesh.MshFileVersion=4.1; <fc-oogmsh>/meshes/condenser11-25.msh -o ...
       <fc-oogmsh>/meshes/condenser11-25-part5.msh' [Gmsh 4.2.2, 1 node, max. 1 thread]
Info    : Started on Sun Mar 24 07:58:03 2019
Info    : Reading '<fc-oogmsh>/meshes/condenser11-25.msh'...
Info    : 3080 nodes
Info    : 6262 elements
Info    : Done reading '<fc-oogmsh>/meshes/condenser11-25.msh'
Info    : Meshing 1D...
Info    : Done meshing 1D (2.6e-05 s)
Info    : Meshing 2D...
Info    : Done meshing 2D (2.6e-05 s)
Info    : 3089 vertices 6311 elements
Info    : Partitioning mesh...
Info    : Running METIS graph partitioner
Info    : 5 partitions, 165 total edge-cuts
Info    : Done partitioning mesh (0.01841 s)
Info    :  - Repartition of 49 point(s): 6(min) 19(max) 9.8(avg)
Info    :  - Repartition of 360 line(s): 68(min) 76(max) 72(avg)
Info    :  - Repartition of 5902 triangle(s): 1180(min) 1181(max) 1180.4(avg)
Info    : Creating partition topology...
Info    :  - Creating partition edges
Info    :  - Creating partition vertices
Info    : Done creating partition topology (0.000342 s)
Info    : Writing '<fc-oogmsh>/meshes/condenser11-25-part5.msh'...
Info    : Done writing '<fc-oogmsh>/meshes/condenser11-25-part5.msh'
Info    : Stopped on Sun Mar 24 07:58:03 2019

[fc-oogmsh] Using gmsh 4.2.2 to write MSH file format version 4.1 in <fc-oogmsh>/meshes/condenser11-25-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.

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.

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 $[0, \texttt{Lx}] \times [0, \texttt{Ly}]$ 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 $[0, \texttt{Lx}] \times [0, \texttt{Ly}]$ 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 ...

Octave 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] Overwritting mesh file <fc-oogmsh>/meshes/rectanglepart-Lx1.000-Ly1.000-Nx3-Ny2-N100.msh
[fc-oogmsh] Starting building mesh <fc-oogmsh>/meshes/rectanglepart-Lx1.000-Ly1.000-Nx3-Ny2-N100.msh with gmsh 4.2.2
[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 4.2.2 to write MSH file format version 4.1 in <fc-oogmsh>/meshes/rectanglepart-Lx1.000-Ly1.000-Nx3-Ny2-N100.msh
```

Octave 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 4.2.2
[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 4.2.2, 1 node, max. 1 thread]
Info    : Started on Sun Mar 24 07:58:05 2019
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    : Meshing curve 1 (Line)
Info    : Meshing curve 2 (Line)
Info    : Meshing curve 3 (Line)
Info    : Meshing curve 4 (Line)
Info    : Meshing curve 5 (Line)
Info    : Meshing curve 6 (Line)
Info    : Meshing curve 7 (Line)
Info    : Meshing curve 8 (Line)
Info    : Meshing curve 9 (Line)
Info    : Meshing curve 10 (Line)
Info    : Meshing curve 11 (Line)
Info    : Meshing curve 12 (Line)
Info    : Meshing curve 13 (Line)
Info    : Meshing curve 14 (Line)
Info    : Meshing curve 15 (Line)
Info    : Meshing curve 16 (Line)
Info    : Meshing curve 17 (Line)
Info    : Done meshing 1D (0.004323 s)
Info    : Meshing 2D...
Info    : Meshing surface 19 (Plane, Delaunay)
Info    : Meshing surface 21 (Plane, Delaunay)
Info    : Meshing surface 23 (Plane, Delaunay)
Info    : Meshing surface 25 (Plane, Delaunay)
Info    : Meshing surface 27 (Plane, Delaunay)
Info    : Meshing surface 29 (Plane, Delaunay)
Info    : Done meshing 2D (0.260097 s)
Info    : 13685 vertices 27682 elements
Info    : Writing './toto.msh'...
Info    : Done writing './toto.msh'
Info    : Stopped on Sun Mar 24 07:58:05 2019

[fc-oogmsh] Using gmsh 4.2.2 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` ([2], 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 [2], 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 entities (*`$Entities`*), the partitioned entities (*`$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

Octave 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
[fc-oogmsh] Using gmsh 4.2.2 to write MSH file format version 4.1 in <fc-oogmsh>/meshes/condenser-10.msh
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:
            Elements: (1x1 struct)
            Entities: (1x1 struct)
          MeshFormat: (1x1 struct)
               Nodes: (1x1 struct)
   PartitionedEntities: []
       PeriodicLinks: []
       PhysicalNames: []
                 dim: 2 double
            meshfile: (1x110 char)
                  nq: 9116 double
     partitionnedfile: 0 logical
                   q: (2x9116 double)
           toGlobal: (1x9116 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.

Octave 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] Using gmsh 4.2.2 to write MSH file format version 4.1 in <fc-oogmsh>/meshes/condenser-6.msh
 fc_oogmsh.ooGmsh4 with properties:
   [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] 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] MeshFormat : [1 1] struct
       [2] version : [1 3] char
       [2] file_type : [1 1] double
       [2] data_size : [1 1] 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] PartitionedEntities : [0 0] double
   [1] PeriodicLinks : [0 0] double
   [1] PhysicalNames : [0 0] double
   [1] dim : [1 1] double
   [1] meshfile : [1 109] char
   [1] nq : [1 1] double
   [1] partitionnedfile : [1 1] logical
   [1] q : [2 3325] double
   [1] toGlobal : [1 3325] double
```

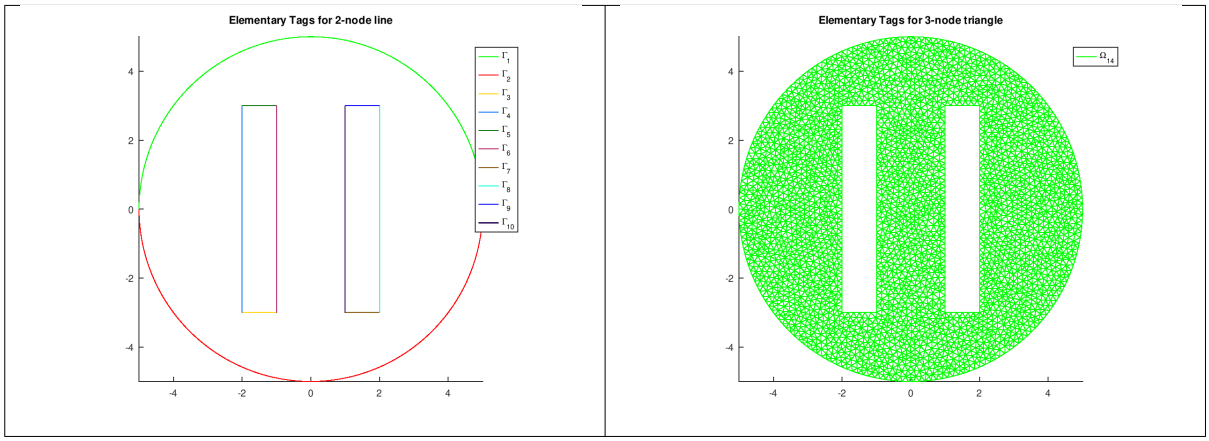


Figure 1: *Elementary Tag* elements of the *geofile* `condenser.geo`

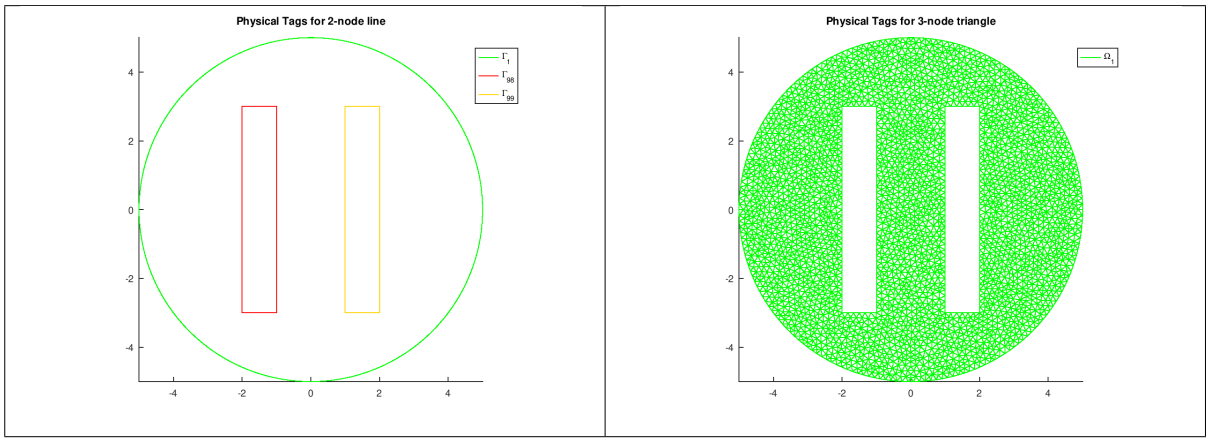


Figure 2: *Physical Tag* elements of the *geofile* `condenser.geo`

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).

Octave 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.

Octave code with output

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

```
phtags1 =

   1   98   99

phtags2 = 1
```

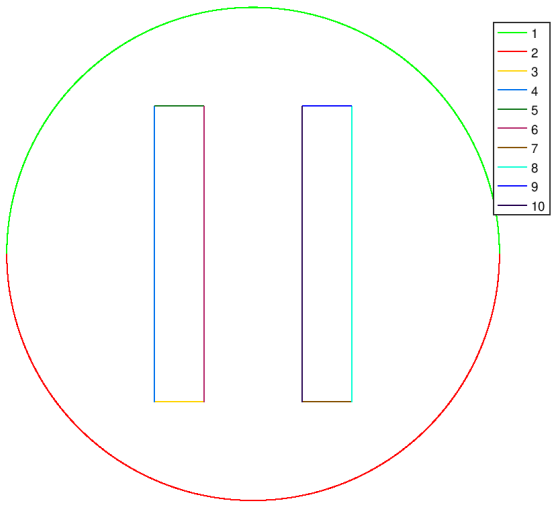
4.1.5 get_me_ElementaryTag method

```
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
fc_tools.graphics.SaveAllFigsAsFiles('ooGmsh4_get_me_ElementaryTags', SaveOptions{:})
```

Listing 1: Plot curves mesh elements by using `get_me_ElementaryTags` function

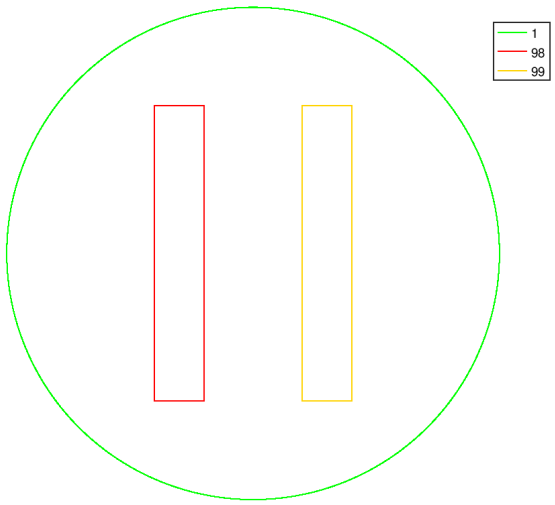
4.1.6 get_me_PhysicalTag method

```
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
fc_tools.graphics.SaveAllFigsAsFiles('ooGmsh4_get_me_PhysicalTags', SaveOptions{:})
```

Listing 2: Plot curves mesh elements by using `get_me_PhysicalTags` function

4.1.7 `get_localmesh_ElementaryTag` method

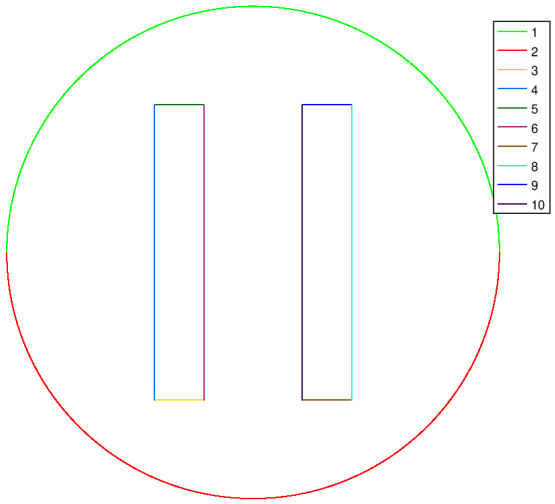
```
[q,me]=Gh.get_localmesh_ElementaryTag(EltType,EltTag)
[q,me,toGlobal]=Gh.get_localmesh_ElementaryTag(EltType,EltTag)
```

`[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
fc_tools.graphics.SaveAllFigsAsFiles('ooGmsh4_get_localmesh_ElementaryTag', SaveOptions{:})
```

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

4.1.8 `get_localmesh_PhysicalTag` method

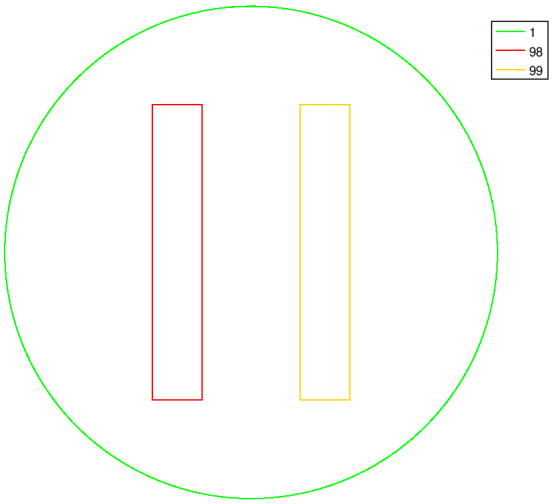
```
[q,me]=Gh.get_localmesh_PhysicalTag(EltType,PhyTag)
[q,me,toGlobal]=Gh.get_localmesh_PhysicalTag(EltType,PhysicalTag)
```

`[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
fc_tools.graphics.SaveAllFigsAsFiles('ooGmsh4_get_localmesh_PhysicalTag', SaveOptions{:})
```

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.
`parametric`	:	integer.
`numNodesBlock`	:	integer.
`nodeTags`	:	1-by-`numNodesBlock` array of integer.
`Nodes`	:	3-by-`numNodesBlock` 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*.

Octave code with output

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

```
[fc-oogmsh] Using gmsh 4.2.2 to write MSH file format version 4.1 in <fc-oogmsh>/meshes/condenser-25.msh
Gh =

  fc_oogmsh.ooGmsh4 with properties:
            Elements: (1x1 struct)
            Entities: (1x1 struct)
          MeshFormat: (1x1 struct)
               Nodes: (1x1 struct)
   PartitionedEntities: []
       PeriodicLinks: []
       PhysicalNames: []
                 dim: 2 double
            meshfile: (1x110 char)
                  nq: 55688 double
     partitionnedfile: 0 logical
                   q: (2x55688 double)
           toGlobal: (1x55688 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*.

Octave code with output

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

```
[fc-oogmsh] Using gmsh 4.2.2 to write MSH file format version 4.1 in <fc-oogmsh>/meshes/cylinderkey-10.msh
Gh =

  fc_oogmsh.ooGmsh4 with properties:
            Elements: (1x1 struct)
            Entities: (1x1 struct)
          MeshFormat: (1x1 struct)
               Nodes: (1x1 struct)
   PartitionedEntities: []
        PeriodicLinks: []
        PhysicalNames: []
                 dim: 3 double
            meshfile: (1x112 char)
                  nq: 5152 double
     partitionnedfile: 0 logical
                   q: (3x5152 double)
             toGlobal: (1x5152 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*.

Octave code with output

```
meshfile=fc_oogmsh.gmsh.buildmesh3ds('ball8',50, 'verbose',0,'force',true);
Gh = fc_oogmsh.ooGmsh4(meshfile)
```

```
[fc-oogmsh] Using gmsh 4.2.2 to write MSH file format version 4.1 in <fc-oogmsh>/meshes/ball8-50.msh
Gh =

  fc_oogmsh.ooGmsh4 with properties:
            Elements: (1x1 struct)
            Entities: (1x1 struct)
          MeshFormat: (1x1 struct)
               Nodes: (1x1 struct)
   PartitionedEntities: []
        PeriodicLinks: []
        PhysicalNames: []
                 dim: 3 double
            meshfile: (1x106 char)
                  nq: 37245 double
     partitionnedfile: 0 logical
                   q: (3x37245 double)
             toGlobal: (1x37245 double)
```

5 ooGmsh2 class (version 2.2)

The ooGmsh class can be used to read `gmsh` mesh files with the MSH ASCII file format (version 2.2) described for example in [1], 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 [1], 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 : $`nb_parts`$(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 `ooGms2` constructor

The `ooGmsh2` class have only one constructor :

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

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

Octave 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
[fc-oogmsh] Using gmsh 4.2.2 to write MSH file format version 2.2 in <fc-oogmsh>/meshes/disk3holes-15.msh
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:
        MeshFormat: (1x1 struct)
               dim: 2 double
          meshfile: (1x111 char)
                nq: 960 double
            orders: 1 double
   partitionnedfile: 0 logical
                 q: (2x960 double)
             sElts: (2x1 struct)
          toGlobal: (1x960 double)
             types: [ 1 2 ] (1x2 double)
```

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.

Octave 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] Using gmsh 4.2.2 to write MSH file format version 2.2 in <fc-oogmsh>/meshes/disk3holes-15.msh
  fc_oogmsh.ooGmsh2 with properties:
    [1] MeshFormat : [1 1] struct
        [2] version : [1 3] char
        [2] file_type : [1 1] double
        [2] data_size : [1 1] double
    [1] dim : [1 1] double
    [1] meshfile : [1 111] char
    [1] nq : [1 1] double
    [1] orders : [1 1] double
    [1] partitionnedfile : [1 1] logical
    [1] q : [2 960] double
    [1] sElts : [2 1] struct
        [2] type : [1 1] double
        [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 : [1 960] double
    [1] types : [1 2] double
```

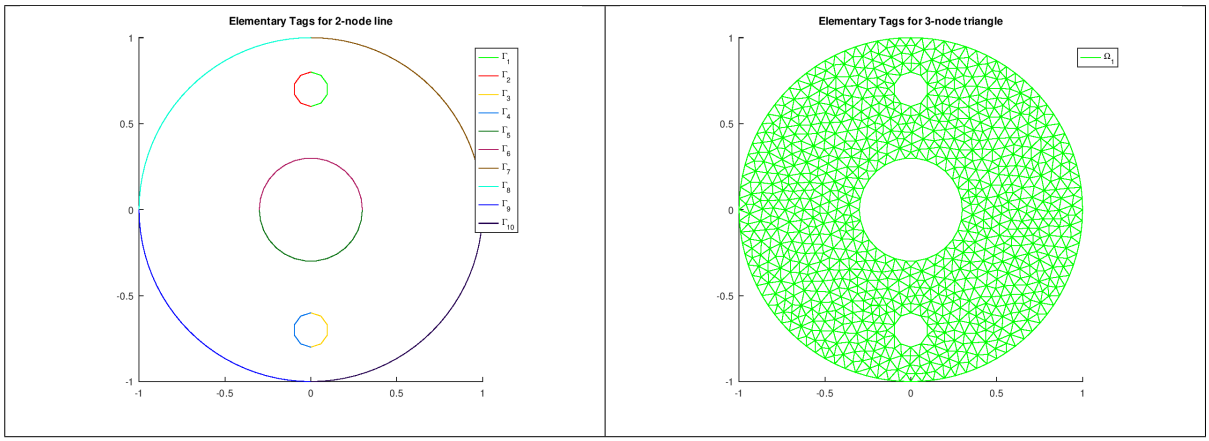


Figure 3: *Elementary Tag* elements of the *geofile* `disk3holes.geo`

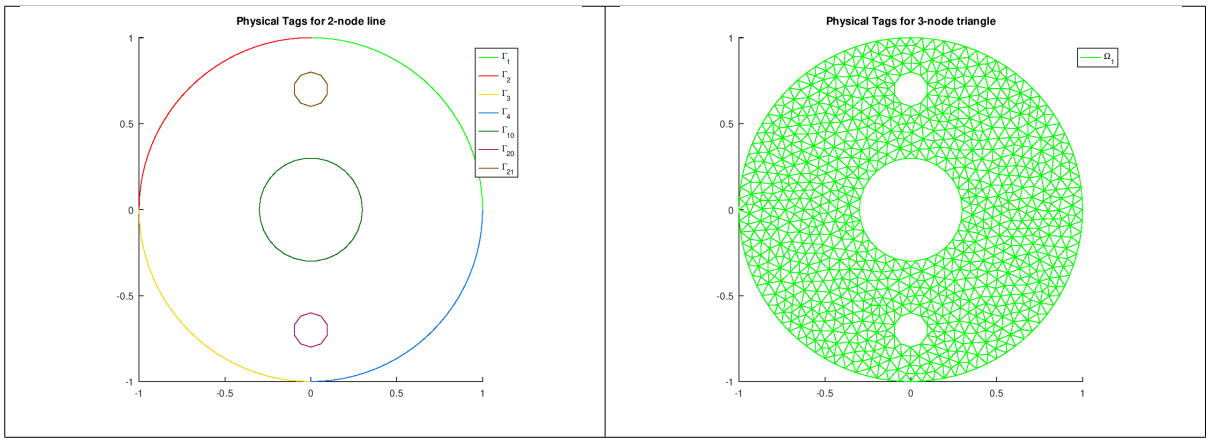


Figure 4: *Physical Tag* elements of the *geofile* `disk3holes.geo`

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).

Octave code with output

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

```
[fc-oogmsh] Using gmsh 4.2.2 to write MSH file format version 2.2 in <fc-oogmsh>/meshes/disk3holes-15.msh
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.

Octave code with output

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

```
[fc-oogmsh] Using gmsh 4.2.2 to write MSH file format version 2.2 in <fc-oogmsh>/meshes/disk3holes-15.msh
phtags1 =

   1   2   3   4  10  20  21

phtags2 = 1
```

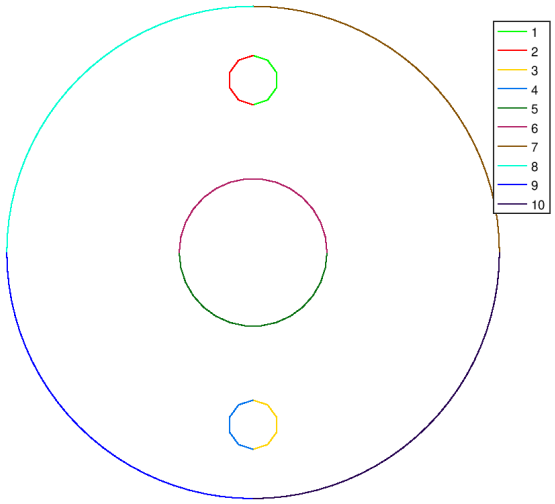
5.1.5 `get_me_ElementaryTag` method

```
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
fc_tools.graphics.SaveAllFigsAsFiles('ooGmsh2_get_me_ElementaryTags', SaveOptions{:})
```

Listing 5: Plot curves mesh elements by using `get_me_ElementaryTags` function

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
fc_tools.graphics.SaveAllFigsAsFiles('ooGmsh2_get_me_PhysicalTags', SaveOptions{:})
```

Listing 6: Plot curves mesh elements by using `get_me_PhysicalTags` function

5.1.7 `get_localmesh_ElementaryTag` method

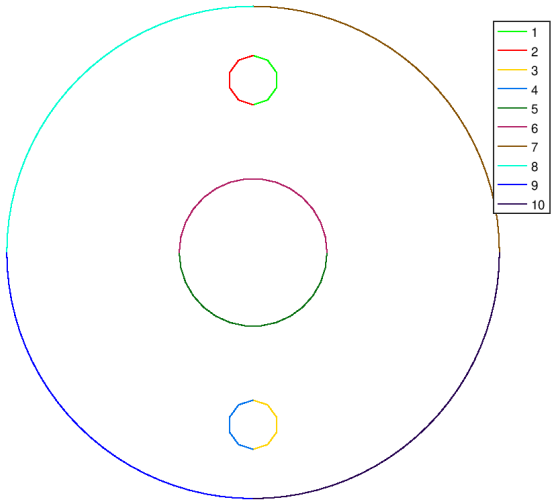
```
[q,me]=Gh.get_localmesh_ElementaryTag(EltType,EltTag)
[q,me,toGlobal]=Gh.get_localmesh_ElementaryTag(EltType,EltTag)
```

`[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`.




```
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
fc_tools.graphics.SaveAllFigsAsFiles('ooGmsh2_get_localmesh_ElementaryTag', SaveOptions{:})
```

Listing 7: Plot curves mesh elements by using `get_localmesh_ElementaryTag` function

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
fc_tools.graphics.SaveAllFigsAsFiles('ooGmsh2_get_localmesh_PhysicalTag', SaveOptions{:})
```

Listing 8: Plot curves mesh elements by using `get_localmesh_PhysicalTag` function

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)
```

```
[fc-oogmsh] Using gmsh 4.2.2 to write MSH file format version 2.2 in <fc-oogmsh>/meshes/condenser-25.msh
Gh =

  fc_oogmsh.ooGmsh2 with properties:
        MeshFormat: (1x1 struct)
               dim: 2 double
          meshfile: (1x110 char)
                nq: 55688 double
            orders: 1 double
   partitionnedfile: 0 logical
                 q: (2x55688 double)
             sElts: (3x1 struct)
         toGlobal: (1x55688 double)
             types: [ 1 2 15 ] (1x3 double)
```

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,'MshFileVersi
Gh = fc_oogmsh.ooGmsh2(meshfile)
```

```
[fc-oogmsh] Using gmsh 4.2.2 to write MSH file format version 2.2 in <fc-oogmsh>/meshes/cylinderkey-10.msh
Gh =

  fc_oogmsh.ooGmsh2 with properties:
        MeshFormat: (1x1 struct)
               dim: 3 double
          meshfile: (1x112 char)
                nq: 5152 double
            orders: 1 double
    partitionnedfile: 0 logical
                 q: (3x5152 double)
            sElts: (3x1 struct)
          toGlobal: (1x5152 double)
             types: [ 1 2 4 ] (1x3 double)
```

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

```
[fc-oogmsh] Using gmsh 4.2.2 to write MSH file format version 2.2 in <fc-oogmsh>/meshes/ball8-50.msh
Gh =

  fc_oogmsh.ooGmsh2 with properties:
        MeshFormat: (1x1 struct)
               dim: 3 double
          meshfile: (1x106 char)
                nq: 37245 double
            orders: 1 double
    partitionnedfile: 0 logical
                 q: (3x37245 double)
            sElts: (3x1 struct)
          toGlobal: (1x37245 double)
             types: [ 1 2 15 ] (1x3 double)
```

A Element type

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

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

Octave 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 =

  scalar structure containing the fields:

    elm_type = 2
    desc = 3-node triangle
    nb_nodes = 3
    order = 1
    incomplete = 0
    d =  2
    geo = triangle

elt4 =

  scalar structure containing the fields:

    elm_type = 4
    desc = 4-node tetrahedron
    nb_nodes = 4
    order = 1
    incomplete = 0
    d =  3
    geo = tetrahedron

elt11 =

  scalar structure containing the fields:

    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
```

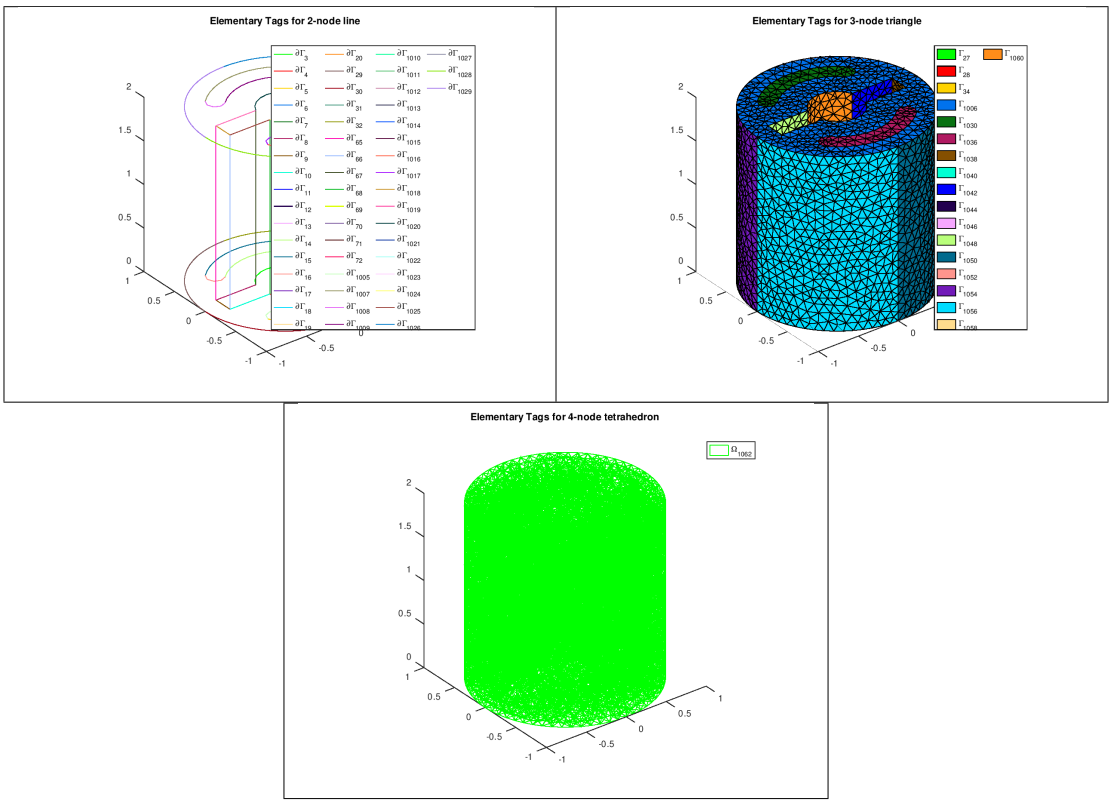
B.2 function fc_oogmsh.plot_ElementaryTags

This function plot *Elementary 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 package [3] version 0.0.4.

```
fc_oogmsh.plot_ElementaryTags(Gh)
```




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

Listing 9: Using `fc_oogmsh.plot_ElementaryTags` function

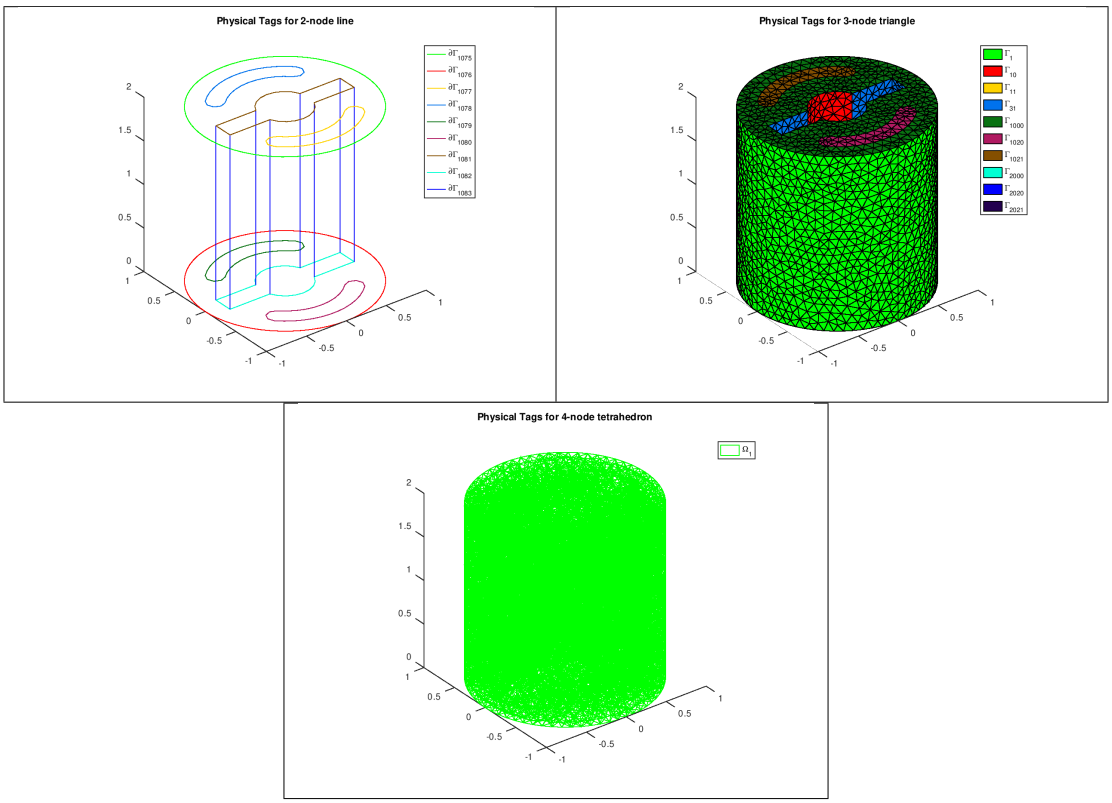
B.3 function `fc_oogmsh.plot_PhysicalTags`

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` package [3] version 0.0.4.

```
fc_oogmsh.plot_PhysicalTags(Gh)
```




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

Listing 10: Using `fc_oogmsh.plot_PhysicalTags` function

B References

[1] Gmsh 2.15.0. `http://gmsh.info`, 2016.

[2] Gmsh 4.2.1. `http://gmsh.info`, 2019.

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

[4] 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.