



FC_OOGMSH package, User's Guide *

François Cuvelier[†]

March 6, 2017

Abstract

This Python package make it possible to generate mesh files from *.geo* file by using `gmsh`. It's also possible with the `ooGMSH` class to read the mesh file and to store its contains in more user-friendly form. This package must be regarded as a very simple interface between `gmsh` files and Python. So you are free to create any data structures or objects you want from an `ooGMSH` object.

Contents

1	Introduction	2
2	gmsh installation (Linux 64bit)	2
3	gmsh interface	3
3.1	function <code>gmsh.buildmesh2d</code>	3
3.2	function <code>gmsh.buildmesh3d</code>	5
3.3	function <code>gmsh.buildmesh3ds</code>	6
3.4	function <code>gmsh.buildpartmesh2d</code>	6
3.5	function <code>gmsh.buildpartmesh3d</code>	7
3.6	function <code>gmsh.buildpartmesh3ds</code>	7
3.7	function <code>gmsh.buildPartRectangle</code>	8

*Compiled with Python 3.6.0, packages `FC_OOGMSH-0.0.2` and `fc_tools-0.0.5`

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

This work was partially supported by ANR Dedales.

4	ooGmsh class	10
4.1	Sample 1	13
4.2	Sample 2	14
4.3	Sample 3	14

1 Introduction

The `FC_OOGMSH` Python package is closely related to `gmsh`, see [2] 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 initially created to make it possible from Python to rapidly

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

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

In a first step we quickly present the installation of `gmsh` on Linux (for Windows and Mac OS X precompiled software are also provided on <http://gmsh.info>). Then, we explain how to configure the `FC_OOGMSH` package for using `gmsh`. Finally, we describe the `FC_OOGMSH`'s functions which use `gmsh` to create mesh files.

remark 1.1

Under Windows the 32bit version of `gmsh` is needed : the system function of Python seems not to support running 64bit applications.

2 gmsh installation (Linux 64bit)

Binaries for Linux 64 bits are available on <http://gmsh.info>).

- To obtain the current stable release (version 2.15.0, December 4 2016) directly from a terminal one can run

```
Terminal
wget http://gmsh.info/bin/Linux/gmsh-2.15.0-
Linux64.tgz
```

- To obtain the development version (automated nightly snapshots) directly from a terminal one can run

Terminal

```
wget http://gmsh.info/bin/Linux/gmsh-svn-Linux64.tgz
```

To install the version 2.15.0 on directory `~/software/GMSH`, one can do the commands

Terminal

```
mkdir -p ~/software/GMSH
cd ~/software/GMSH
wget http://gmsh.info/bin/Linux/gmsh-2.15.0-Linux64.tgz
tar zxvf gmsh-2.15.0-Linux64.tgz
```

To run `gmsh` we can use the following command in a terminal

Terminal

```
~/software/GMSH/gmsh-2.15.0-Linux/bin/gmsh &
```

3 gmsh interface

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

3.1 function `gmsh.buildmesh2d`

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

Syntaxe

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

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

Description

`meshfile=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=gmesh.buildmesh2d(geofile,N,Key=Value, ...)` specifies function options using one or more Key,Value pair arguments. The Key options can be

- **meshdir** : to specify the directory where the mesh file will be written, (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; ...)
- **options** : string which contains command-line options used with gmsh (see gmsh documentation)

Examples All the following examples use the *.geo* file `condenser11.geo` which is in the directory `geodir` of the toolbox.

Python commands with output

```
print( '***_gmsh.buildmesh2d_:1st_call ' )
meshfile=gmesh.buildmesh2d( 'condenser11 ',25,force=True)
print( '***_gmsh.buildmesh2d_:2nd_call ' )
meshfile=gmesh.buildmesh2d( 'condenser11 ',25)
```

```
*** gmsh.buildmesh2d : 1st call
Try to use default parameters!
Use fc_oogmesh.configure to configure.

-> Using GMSH binary : /home/cuvelier/bin/gmsh

Write in /fcopt/python3.6.0/lib/python3.6/site-packages/configure_loc.py ...

-> done

Running gmsh. Be patient...
-> done!
*** gmsh.buildmesh2d : 2nd call
Mesh file /fcopt/python3.6.0/lib/python3.6/site-packages/meshes/condenser11-25.msh already exist.
-> Use "force" flag to rebuild if needed.
```

Python commands with output

```
meshfile=gmsht.buildmesh2d('condenser11',25,force=True,
    verbose=True, options='-string Mesh.Algorithm=1;-string
    -string Mesh.ScalingFactor=2;')
```

```
Overwriting mesh file /fcpt/python3.6.0/lib/python3.6/site-packages/meshes/condenser11-25.msh
Running gmsht. Be patient...
Info : Running '/home/cuvellier/bin/gmsh -2 -setnumber N 25 -string Mesh.Algorithm=1; -string Mesh.ScalingFactor=2;
      /fcpt/python3.6.0/lib/python3.6/site-packages/fc_oogmsh/geodir/2d/condenser11.geo -o
      /fcpt/python3.6.0/lib/python3.6/site-packages/meshes/condenser11-25.msh' [Gmsh 2.16.0, 1 node, max. 1 thread]
Info : Started on Mon Mar 6 18:33:11 2017
Info : Reading '/fcpt/python3.6.0/lib/python3.6/site-packages/fc_oogmsh/geodir/2d/condenser11.geo'...
Info : Reading '/fcpt/python3.6.0/lib/python3.6/site-packages/fc_oogmsh/geodir/2d/options01_data.geo'...
Info : Done reading '/fcpt/python3.6.0/lib/python3.6/site-packages/fc_oogmsh/geodir/2d/options01_data.geo'
Info : Reading '/fcpt/python3.6.0/lib/python3.6/site-packages/fc_oogmsh/geodir/2d/shape_functions.geo'...
Info : Done reading '/fcpt/python3.6.0/lib/python3.6/site-packages/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 '/fcpt/python3.6.0/lib/python3.6/site-packages/fc_oogmsh/geodir/2d/condenser11.geo'
Info : Finalized high order topology of periodic connections
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 (0.010406 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 (0.397911 s)
Info : 2999 vertices 6131 elements
Info : Writing '/fcpt/python3.6.0/lib/python3.6/site-packages/meshes/condenser11-25.msh'...
Info : Done writing '/fcpt/python3.6.0/lib/python3.6/site-packages/meshes/condenser11-25.msh'
Info : Stopped on Mon Mar 6 18:33:12 2017
-> done!
```

3.2 function gmsht.buildmesh3d

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

3.3 function 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 gmsh.buildpartmesh2d

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

Syntaxe

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

Description

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

- **'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; ...)
- **options** : string which contains command-line options used with `gmsh` (see `gmsh` documentation)

Examples All the following examples use the `meshfile` as output of the command :

```
meshfile=gmsh.buildmesh2d('condenser11',25);
```

Python commands with output

```
print('***_Building_the_mesh')
meshfile=gmsht.buildmesh2d('condenser11',25);
print('***_Partitioning_the_mesh')
pmlfile=gmsht.buildpartmesh(meshfile,5,force=True);
```

```
*** Building the mesh
Mesh file /fcopt/python3.6.0/lib/python3.6/site-packages/meshes/condenser11-25.msh already exist.
-> Use "force" flag to rebuild if needed.
*** Partitioning the mesh
Running gmsht. Be patient...
-> done!
```

Python commands with output

```
print('***_Building_the_mesh')
meshfile=gmsht.buildmesh2d('condenser11',25, verbose=True);
print('***_Partitioning_the_mesh')
pmlfile=gmsht.buildpartmesh(meshfile,5,force=True,
    verbose=True, options='-string "Mesh.Partitioner=2;"_
    -string "Mesh.MetisAlgorithm=3;"')
```

```
*** Building the mesh
Mesh file /fcopt/python3.6.0/lib/python3.6/site-packages/meshes/condenser11-25.msh already exist.
-> Use "force" flag to rebuild if needed.
*** Partitioning the mesh
Overwriting mesh file /fcopt/python3.6.0/lib/python3.6/site-packages/meshes/condenser11-25-part5.msh
Running gmsht. Be patient...
Info : Running '/home/cuvellier/bin/gmsht -2 -saveall -part 5 -string Mesh.Partitioner=2; -string Mesh.MetisAlgorithm=3;
      /fcopt/python3.6.0/lib/python3.6/site-packages/meshes/condenser11-25.msh -o
      /fcopt/python3.6.0/lib/python3.6/site-packages/meshes/condenser11-25-part5.msh' [Gmsht 2.16.0, 1 node, max. 1
      thread]
Info : Started on Mon Mar 6 18:33:12 2017
Info : Reading '/fcopt/python3.6.0/lib/python3.6/site-packages/meshes/condenser11-25.msh'...
Info : 2990 vertices
Info : 6082 elements
Info : Done reading '/fcopt/python3.6.0/lib/python3.6/site-packages/meshes/condenser11-25.msh'
Info : Finalized high order topology of periodic connections
Info : Meshing 1D...
Info : Done meshing 1D (3.2e-05 s)
Info : Meshing 2D...
Info : Done meshing 2D (2.09808e-05 s)
Info : 2990 vertices 6082 elements
Info : Building graph...
Info : Partitioning graph...
Info : Launching METIS graph partitioner
METIS with weights
Info : Number of Edges Cut : 133
Info : Done partitioning graph
Info : Writing '/fcopt/python3.6.0/lib/python3.6/site-packages/meshes/condenser11-25-part5.msh'...
Info : Done writing '/fcopt/python3.6.0/lib/python3.6/site-packages/meshes/condenser11-25-part5.msh'
Info : Stopped on Mon Mar 6 18:33:12 2017
-> done!
```

3.5 function gmsht.buildpartmesh3d

This function uses `gmsht` and a `.msh` file (containing of a 3D-mesh) to generate a 3D partioned mesh.

3.6 function gmsht.buildpartmesh3ds

This function uses `gmsht` and a `.msh` file (containing of a 3D surface mesh) to generate a 3D partioned surface mesh.

3.7 function gmsh.buildPartRectangle

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

Syntaxe

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

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

Description

`meshfile=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 : `rectanglepart-Lx%.3f-Ly%.3f-Nx%d-Ny%d-N%d.msh`

`meshfile=gmsh.buildpartrectangle(Lx,Ly,Nx,Ny,N,Key=Value, ...)` specifies function options using one or more `Key, Value` pair arguments. The `Key` options can be

- `'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; ...)
- `options` : string which contains command-line options used with `gmsh` (see `gmsh` documentation)

Examples All the following examples ...

Python commands with output

```
pmfile=gmshtool.buildpartrectangle(1,1,3,2,100,force=True,
verbose=True)
```

```
Overwritting mesh file rectanglepart-Lx1.000-Ly1.000-Nx3-Ny2-N100.msh
Running gmshtool. Be patient...
Info : Running '/home/cuvellier/bin/gmsh -2 -setnumber N 100 -setnumber NX 3 -setnumber NY 2 -setnumber LX 1 -setnumber
LY 1 /fcopt/python3.6.0/lib/python3.6/site-packages/fc_oogmsh/geodir/2d/rectanglepart.geo -o
rectanglepart-Lx1.000-Ly1.000-Nx3-Ny2-N100.msh' [Gmsh 2.16.0, 1 node, max. 1 thread]
Info : Started on Mon Mar 6 18:33:13 2017
Info : Reading '/fcopt/python3.6.0/lib/python3.6/site-packages/fc_oogmsh/geodir/2d/rectanglepart.geo'...
Info : Reading '/fcopt/python3.6.0/lib/python3.6/site-packages/fc_oogmsh/geodir/2d/partitions01.data.geo'...
Info : Done reading '/fcopt/python3.6.0/lib/python3.6/site-packages/fc_oogmsh/geodir/2d/partitions01.data.geo'
Info : Reading '/fcopt/python3.6.0/lib/python3.6/site-packages/fc_oogmsh/geodir/2d/partitions_shape.geo'...
Info : Done reading '/fcopt/python3.6.0/lib/python3.6/site-packages/fc_oogmsh/geodir/2d/partitions_shape.geo'
Info : Done reading '/fcopt/python3.6.0/lib/python3.6/site-packages/fc_oogmsh/geodir/2d/rectanglepart.geo'
Info : Finalized high order topology of periodic connections
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 (0.002417 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 (0.516607 s)
Info : 13685 vertices 27682 elements
Info : Writing 'rectanglepart-Lx1.000-Ly1.000-Nx3-Ny2-N100.msh'...
Info : Done writing 'rectanglepart-Lx1.000-Ly1.000-Nx3-Ny2-N100.msh'
Info : Stopped on Mon Mar 6 18:33:13 2017
-> done!
```

Python commands with output

```
pmfile=gmshtool.buildpartrectangle(1,1,3,2,100,verbose=True,
force=True, meshfile='./toto.msh')
```

```
Overwriting mesh file ./toto.msh
Running gmsh. Be patient...
Info : Running '/home/cuvellier/bin/gmsh -2 -setnumber N 100 -setnumber NX 3 -setnumber NY 2 -setnumber LX 1 -setnumber
LY 1 /fcpt/python3.6.0/lib/python3.6/site-packages/fc_oogmsh/geodir/2d/rectanglepart.geo -o ./toto.msh' [Gmsh
2.16.0, 1 node, max. 1 thread]
Info : Started on Mon Mar 6 18:33:13 2017
Info : Reading '/fcpt/python3.6.0/lib/python3.6/site-packages/fc_oogmsh/geodir/2d/rectanglepart.geo'...
Info : Reading '/fcpt/python3.6.0/lib/python3.6/site-packages/fc_oogmsh/geodir/2d/partitions01.data.geo'...
Info : Done reading '/fcpt/python3.6.0/lib/python3.6/site-packages/fc_oogmsh/geodir/2d/partitions01.data.geo'
Info : Reading '/fcpt/python3.6.0/lib/python3.6/site-packages/fc_oogmsh/geodir/2d/partitions_shape.geo'...
Info : Done reading '/fcpt/python3.6.0/lib/python3.6/site-packages/fc_oogmsh/geodir/2d/partitions_shape.geo'
Info : Done reading '/fcpt/python3.6.0/lib/python3.6/site-packages/fc_oogmsh/geodir/2d/rectanglepart.geo'
Info : Finalized high order topology of periodic connections
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 (0.003032 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 (0.524523 s)
Info : 13685 vertices 27682 elements
Info : Writing './toto.msh'...
Info : Done writing './toto.msh'
Info : Stopped on Mon Mar 6 18:33:14 2017
-> done!
```

4

ooGmsh class

The **ooGMSH** class can be used to read **gmsh** mesh files with the MSH ASCII file format described for example in [1], section 9.1.

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

<i>elm-type</i>	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)
----	---

When reading a .msh file 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 different *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 <i>elm-type</i> description of [1], section 9.1.
geo	: string contains the kind of geometry: 'line', 'triangle', 'tetrahedron', ...
d	: integer space dimension or <i>d</i> -simplex.
order	: integer order of the element
n _{me}	: integer number of mesh elements
me	: array of <i>d</i> + 1-by-n _{me} integers connectivity array
phys_lab	: array of n _{me} -by-... integers physical labels of the elements
geo_lab	: array of n _{me} -by-... integers geometrical labels of the elements
nb_parts	: array of n _{me} -by-1 integers number of mesh partitions to which the element belongs
part_lab	: array of n _{me} -by-max(nb_parts) integers part_lab(<i>i</i> , 1 : nb_parts(<i>i</i>)) contains all the partitions index to which the <i>i</i> -th element belongs.

The `ooGmsh` 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>i</i>) contains all the elements of type types(<i>i</i>) and order orders(<i>i</i>).

The **ooGmsh** class have only one constructor :

```
Gh=ooGmsh(meshfile)
```

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

4.1 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*.

Python commands with output

```
meshfile=gmesh.buildmesh2d('condenser',25);
Gh = gmsh.ooGmsh(meshfile)
print('***_Gh:')
print(Gh)
print('***_Gh.sElts[0]:')
print(Gh.sElts[0])
```

```
Running gmsh. Be patient...
-> done!
*** Gh:
ooGmsh object
  dim : 2
  types : [ 1 2 15]
  orders : [1]
  nq : 55670
  q : ndarray object[float64], size (55670, 2)
  toGlobal: ndarray object[int64], size (55670,)
  sElts : list of 3 elements
*** Gh.sElts[0]:
ooMs object
  d : 1, type : 1, order : 1
  nme : 1486
  me : ndarray object[int64], size (2, 1486)
  phys_lab: ndarray object[int64], size (1486,)
  geo_lab : ndarray object[int64], size (1486,)
  part_lab: list of 1486 elements
  nb_parts: ndarray object[int64], size (1486,)
  nTags : ndarray object[int64], size (1486,)
```

4.2 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*.

Python commands with output

```
meshfile=gmsht.buildmesh3d('cylinderkey',10, force=True)
Gh = gmsht.ooGmsht(meshfile)
print('***_Gh:')
print(Gh)
print('***_Gh.sElts[1]:')
print(Gh.sElts[1])
```

```
Running gmsht. Be patient...
-> done!
*** Gh:
ooGmsht object
dim : 3
types : [1 2 4]
orders : [1]
nq : 5129
q : ndarray object[float64], size (5129, 3)
toGlobal: ndarray object[int64], size (5129,)
sElts : list of 3 elements
*** Gh.sElts[1]:
ooMs object
d : 2, type : 2, order : 1
nme : 6512
me : ndarray object[int64], size (3, 6512)
phys_lab: ndarray object[int64], size (6512,)
geo_lab : ndarray object[int64], size (6512,)
part_lab: list of 6512 elements
nb_parts: ndarray object[int64], size (6512,)
nTags : ndarray object[int64], size (6512,)
```

4.3 Sample 3

The 3d .geo file *sphere8surf.geo* is used to create a 3d surface .msh file : *sphere8surf-40.msh*. This .msh file contains 1 (2-node line), 2 (3-node triangle) and 15 (1-node point) *elm-type*.

Python commands with output

```
meshfile=gmesh.buildmesh3ds('sphere8surf',40,force=True)
Gh = gmsh.ooGmsh(meshfile)
print('***_Gh: ')
print(Gh)
print('***_Gh.sElts[0]: ')
print(Gh.sElts[0])
```

```
Running gmsh. Be patient...
-> done!
*** Gh:
ooGmsh object
  dim : 3
  types : [ 1 2 15]
  orders : [1]
  nq : 23312
  q : ndarray object[float64], size (23312, 3)
toGlobal: ndarray object[int64], size (23312,)
sElts : list of 3 elements
*** Gh.sElts[0]:
ooMs object
  d : 1, type : 1, order : 1
  nme : 756
  me : ndarray object[int64], size (2, 756)
  phys_lab : ndarray object[int64], size (756,)
  geo_lab : ndarray object[int64], size (756,)
  part_lab : list of 756 elements
  nb_parts : ndarray object[int64], size (756,)
  nTags : ndarray object[int64], size (756,)
```

References

- [1] Gmsh 2.15.0. <http://gmsh.info>, 2016.
- [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.