

Software Tool for Automation of Pre-Processing in the Finite Element/Volume Method

User's Guide

Author:

Filip Zaoral

Ostrava, November 2022

Table of Contents

Table of Contents	2
1 Description	3
2 Structure of the library	3
3 Installation	4
3.1 In Windows operating system	4
3.2 In Linux operating system	5
4 Getting started	5
4.1 The first way.....	5
4.2 The second way	5
4.3 After solution.....	6
5 Input Geometry	6
5.1 Geometry in the IGES Format.....	7
5.2 Geometry in the STL Format	7
6 GCF Configuration File	8
6.1 General parameters.....	9
6.2 STL topology parameters	10
6.3 Global meshing parameters	11
6.4 Local meshing parameters.....	14
6.5 Inflation layers parameters	15
Appendix A Structure of the MSH File	17

1 Description

Purpose of this library is to automatize the preparation of finite element computational models. In addition to the discretization of primarily three-dimensional geometry in IGES or STL format into finite elements/volumes, it also allows for automatic assignment of elements and nodes into groups, corresponding to geometric entities (surfaces only for now) that have been assigned an arbitrary name (not containing dots) during pre-processing in a suitable CAD software package. This library is supported on Windows and Linux operating systems.

2 Structure of the library

In the root directory of the library there are:

- The "lib" directory,
- the "examples" directory with all the test problems created by the author,
- the "doc" directory containing entire documentation for the library,
- a launch file "GMTLaunch.py" to launch the tool,
- the "Gmsh.py" script to launch the Gmsh GUI conveniently,
- a configuration file with the extension .gcf (hereafter GCF), and
- the license file LICENSE.txt.

The "Lib" directory represents the library of the software tool itself. Along with the source code of the tool, it also contains the Gmsh library software development kit. The GCF file contains the user-specified settings for the finite element mesh generator. The LICENSE.txt file contains the exact wording of the GNU GPL version 2 license agreement.

The input model geometry file(s) are placed in the job working directory. The tool also stores its outputs here i.e., the resulting finite element mesh file in user-selected format and a .log (hereafter LOG) file containing informative, warning and error messages through which the tool communicates with the user. The structure of the library is illustrated in Fig. 4.

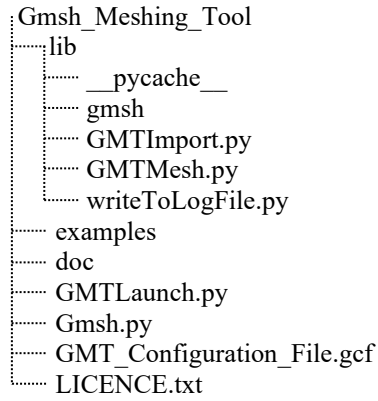


Fig. 1 Structure of the library.

3 Installation

The library is written in Python. To install the tool, one must first have Python installed. Compatibility is guaranteed for Python versions 3.10.x, for other versions, full functionality is not fully guaranteed.

After that, it is required to install the NumPy library. This can be done easily with the `pip3 install numpy` command in the command line in the case of Windows operating system (run as administrator) or terminal in the case of Linux based system.

Next step is to unzip the compressed directory containing the library into the installation directory of the user's choice.

All that remains is to set up an environment variable pointing to the installation directory of the tool.

3.1 In Windows operating system

To set up an environment variable in Windows the standard way:

1. Right click on "This Computer",
2. then select "Properties" from the popup menu,
3. next click on "Advanced System Settings",
4. then on "Advanced" tab,
5. click on "Environment Variables",
6. under "System Variables" click on "New...",
7. then in the "Variable Name:" field, type `GMTPATH` (uppercase), and
8. in the "Variable Value:" field, input the absolute path to the installation directory of the tool.

3.2 In Linux operating system

To set up an environment variable in Linux, first use the `nano ~/.bashrc` command in the terminal to open (or create and edit if none exists) the `.bashrc` file.

Next add the following line `export GMTPATH="absolute path to the installation directory of the tool"`, note that the quotation marks should be used if the path contains spaces.

4 Getting started

To solve a job, this tool can be used in two different ways:

4.1 The first way

To use the tool this way, first create a working directory and place the model geometry files in the appropriate format in it.

Next, set the desired parameter values in the GCF configuration file located in a root directory of the library. It is necessary to set the `WorkingDirectoryPath` parameter with the absolute path to the working directory and `Name` parameter with the name of the model.

Now all that is left to do is to run the file `"GMTLaunch.py"` which is also located in the root directory.

4.2 The second way

Copy the filled GCF file and the `"GMTLaunch.py"` launch file to the working directory containing the model geometry files in the appropriate format and rename those two files there with the model name.

In this case, it is not necessary to set the `WorkingDirectoryPath` and `Name` parameters in the GCF file, as these are filled in automatically when the launch file is executed from the working directory.

4.3 After solution

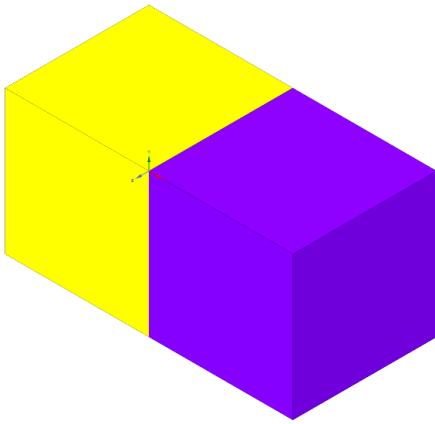
When the job is complete, the tool writes the resulting finite element mesh to a file in the format specified in GCF file and saves it in the job's working directory along with the LOG file. If a fatal problem occurred while solving the job, then the tool saves only the LOG file with a possible explanation of a cause of the problem.

5 Input Geometry

The tool requires two different inputs for its use. The first required input is the model geometry files. Files in IGES (with extension .igs or .iges) or STL (with extension .stl) format are accepted. However, depending on the format used, certain restrictions apply that impose requirements on the arrangement of the geometry files and their names.

Each job of the tool consists of a one or more bodies, volumetric or planar. All bodies that are adjacent to each other form a so-called island, see Fig. 5 a). Each island then consists of one or more bodies and those islands are not connected to each other.

a) Model geometry consisting of a single island made of two adjacent volumes.



b) Model geometry consisting of two islands, each made of single volume

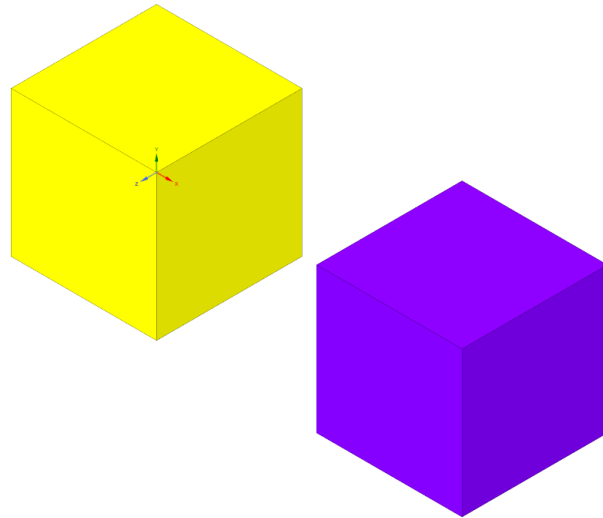


Fig. 2 Model geometry consisting of a single island versus multiple islands.

If the model consists of several disconnected islands, see Fig. 5 b), it is then necessary to submit a job for each island separately. However, each a job is submitted, a separate process is assigned to it and the solution of the entire problem is thus parallelized at the level of said islands.

Each body of each island must then be assigned a separate geometry file or files in the same format (IGES or STL).

5.1 Geometry in the IGES Format

In the case of the IGES format, the complete name of each file consists of the model name common to all model files, the name of the body to which the particular file belongs, and the file extension, always separated by a period, see option a) in Fig. 6 below.

A shortened name, consisting only of the model name and the suffix, is also acceptable, see option b). All bodies in the model will then be assigned the model name, followed by a sequence number, as their name.

It is also possible to split the geometry into IGS files already at the level of shells, see option c) in Fig. 6 below. Each shell consists of a number of faces that are adjacent to each other. In this case, the names of the files corresponding to the shells separating two bodies must then specifically follow the format shown in variant d) in Fig. 6. The file name in this case contains the names of the two bodies separated by the shell.

In any case, however, it must be ensured that the faces in each file are adjacent to each other (analogous to islands of bodies, no disconnected faces allowed).

For models in IGES format, all length dimensions are always expressed in millimetres.

- a) <model name>.<body name>.igs
- b) <model name>.igs
- c) <model name>.<body name>.<shell name>.igs
- d) <model name>.<body 1 name>.<body 2 name>.<shell name>.igs

Fig. 3 Possible variants of the names of the IGS geometry files, brackets <> are not written.

5.2 Geometry in the STL Format

Analogous naming conventions for geometry files with the model name, body name, and the file extension apply also to the STL format.

When the model consists of only one body, variants a) and b) in Fig. 7 below can be used. However, beware that if the STL geometry files contain user-named surfaces, then it is necessary that these files are in ASCII form. This is because the Gmsh library does not yet support reading surface names from binary files.

Otherwise, when the model is composed of multiple bodies, it is necessary to split the geometry into STL files already at the level of shells, see options c) in Fig. 7 below. Each shell consists of a number of faces that are adjacent to each other. The names of the files corresponding to the shells separating two bodies must then specifically follow the format shown in variant d) in Fig. 7. The file name in this case contains the names of the two bodies separated by the shell. In this case, all the geometry files may be in either ASCII or binary form.

In the case of models in STL format, all length dimensions are always expressed in the units in which the model was exported from the geometry pre-processor.

- a) <model name>.<body name>.stl
- b) <model name>.stl
- c) <model name>.<body name>.<shell name>.stl
- d) <model name>.<body 1 name>.<body 2 name>.<shell name>.stl

Fig. 4 Possible variants of the names of the STL geometry files, brackets <> are not written.

6 GCF Configuration File

The second input required for the use of the tool is the GCF configuration file. This is an ASCII file with UTF-8 encoding. It can therefore be opened and modified as a regular text file in any of a plethora of word processing applications.

Each line of the file contains one parameter the value of which can be one of the following data types:

- The "Integer" data type represents whole numbers,
- the "Float" data type represents decimal numbers,
- the "Bool" data type represents logical values, i.e., "True", "False", "Yes", "No", "1", or "0",
- the "String" data type represents strings of Unicode characters (words), and finally
- the "List" data type represents a list of values of the "String" data type, separated by a comma ", " or semicolon ";".

Each line begins with the parameter name, followed by the equals sign "=", which is followed by the parameter value itself. Optionally, informative comments beginning with the "#" character can be added either at the end of the lines or on new separate lines.

Individual parameters are described in detail below. These are divided into several categories, namely general parameters, STL topology parameters, global meshing parameters, local meshing parameters, and parameters for generation of inflation layers.

6.1 General parameters

WorkingDirectoryPath

Absolute path to the working directory.

Data type: String

Possible values:Any

Default value: None

Name

Name of the model/job.

Data type: String

Possible values:Any

Default value: None

InputFormat

Format of input geometry files (1: IGES, 2: STL).

Data type: Integer

Possible values:1 or 2

Default value: 1

MaxNumThreads

Maximum number of CPU threads to use for meshing of surfaces and volumes if the HXT 3D mesher is used (MeshAlgorithm3D = 10).

Data type: Integer

Possible values:Any ≥ 1

Default value: 1

GeometryTolerance

Tolerance of length dimensions, by default calculated from the diagonal of the bounding box of the model geometry d . Physical unit depends on geometry input format, in case of IGES it is [mm], in case of STL it is the same physical unit that the geometry was exported in from the geometry pre-processor.

Data type: Float

Possible values: Any > 0

Default value: $1 \cdot 10^{-9} \cdot d$

OutputFormat

Format of the output mesh files (1: MSH, 2: UNV, 3: VTK).

Data type: Integer

Possible values: 1 to 3

Default value: 1

Binary

Save the mesh to a binary file if possible?

Data type: Bool

Possible values: True or False

Default value: False

LaunchGmsh

Run Gmsh to see the result after finishing the job?

Data type: Bool

Possible values: True or False

Default value: False

6.2 STL topology parameters

STLRemesh

Parameterize and remesh the STL model topology?

Data type: Bool

Possible values: True or False

Default value: False

STLFacetAngle

Minimum angle [°] between the normals of two adjacent triangles at which their common segment is already considered a sharp edge.

Data type: Float

Possible values:Any from 0. to 180.

Default value: 45.

STLCurveAngle

Minimum angle [°] between two adjacent segments at which their common point is already considered a sharp corner.

Data type: Float

Possible values:Any from 0. to 180.

Default value: 180.

6.3 Global meshing parameters

MeshDim

Number of dimensions of the resulting mesh (0: 0D, 1: 1D, 2: 2D, 3: 3D).

Data type: Integer

Possible values:0 to 3

Default value: 3

MeshSizeFromCurvature

Target number of segments per 2π radians of curvature.

Data type: Integer

Possible values:Any ≥ 0

Default value: 6

MeshSizeExtendFromBoundary

Extend computation of mesh element sizes from the boundaries into the interior?

Data type: Bool

Possible values:True or False

Default value: True

MeshAlgorithm

2D meshing algorithm (1: MeshAdapt, 2: Automatic, 3: Initial mesh only, 5: Delaunay, 6: Frontal-Delaunay, 7: BAMG, 8: Frontal-Delaunay for Quads, 9: Packing of Parallelograms).

Data type: Integer

Possible values: 1, 2, 3, 5, 6, 7, 8, or 9

Default value: 6

MeshAlgorithm3D

3D meshing algorithm (1: Delaunay, 3: Initial mesh only, 4: Frontal, 7: MMG3D, 9: R-tree, 10: HXT).

Data type: Integer

Possible values: 1, 3, 4, 7, 9, or 10

Default value: 10

MeshSizeMax

Maximum global size of the elements, by default calculated from the diagonal of the bounding box of the model geometry d . Physical unit depends on geometry input format, in case of IGES it is [mm], in case of STL it is the same physical unit that the geometry was exported in from the geometry pre-processor.

Data type: Float

Possible values: Any > 0 .

Default value: $2 \cdot 10^{-2} \cdot d$

MeshSizeMin

Minimum global size of the elements. Physical unit depends on geometry input format, in case of IGES it is [mm], in case of STL it is the same physical unit that the geometry was exported in from the geometry pre-processor.

Data type: Float

Possible values: Any ≥ 0 .

Default value: 0

MinimumCircleNodes

Minimum number of nodes per circle or ellipse.

Data type: Integer

Possible values: Any ≥ 0

Default value: 0

MinimumCurveNodes

Minimum number of nodes per curve other than a line, circle, or ellipse.

Data type: Integer

Possible values: Any ≥ 0

Default value: 0

ElementOrder

Order of the elements (1: linear, 2: quadratic, 3: cubic).

Data type: Integer

Possible values: Any ≥ 1

Default value: 1

SecondOrderIncomplete

Create incomplete higher order elements (8-node 2nd order quadrilateral, 20-node 2nd order hexahedral, etc.)?

Data type: Bool

Possible values: True or False

Default value: True

SecondOrderLinear

Create mid-side nodes of higher-order elements and nodes resulting from "subdivision" algorithms by simple linear interpolation?

Data type: Bool

Possible values: True or False

Default value: False

Optimize

Optimize the mesh to improve the quality of tetrahedral elements?

Data type: Bool

Possible values: True or False

Default value: True

OptimizeNetgen

Optimize the mesh to improve the quality of tetrahedral elements using the Netgen library?

Data type: Bool

Possible values: True or False

Default value: False

HighOrderOptimize

Algorithm for optimization of higher order element mesh (0: none, 1: optimization, 2: elastic and optimization, 3: elastic, 4: fast curving).

Data type: Integer

Possible values: 0 to 4

Default value: 4

6.4 Local meshing parameters

LocalMeshSurfaces

List of surface names on which to enforce the local element size.

Data type: String

Possible values: Any

Default value: None

LocalMeshVolumes

List of volume names on which to enforce the local element size.

Data type: String

Possible values: Any

Default value: None

LocalMeshSize

Target local size of the elements on the selected surfaces/volumes. Physical unit depends on geometry input format, in case of IGES it is [mm], in case of STL it is the same physical unit that the geometry was exported in from the geometry pre-processor.

Data type: Float

Possible values: Any > 0 .

Default value: None

LocalMeshGrowthRate

Rate of change of size of the neighboring elements near the selected surfaces/volumes.

Data type: Float

Possible values: Any ≥ 0 .

Default value: None

The local meshing parameters can be declared repeatedly for multiple groups of geometric entities (surfaces and/or volumes) with varying values of parameters `LocalMeshSize` and `LocalMeshGrowthRate`, but always together as a foursome.

6.5 Inflation layers parameters

InflationLayersSurfaces

List of surface names on which to generate inflation layers.

Data type: String

Possible values: Any

Default value: None

InflationLayers

Number of inflation layers in normal direction to generate on the selected surfaces.

Data type: Integer

Possible values: Any ≥ 1

Default value: None

InflationLayersMethod

Method of specification of inflation layer thickness (1: Total thickness, 2: First layer thickness, 3: Total aspect ratio, 4: First layer aspect ratio, 5: Last layer transition ratio (experimental)).

Data type: Integer

Possible values: 1 to 5

Default value: None

InflationLayersThickness

Inflation layer thickness parameter whose effect depends on the chosen method of specification:

- If `InflationLayersMethod` = 1 then its value defines total thickness of all layers,
- if `InflationLayersMethod` = 2 then its value defines first layer thickness,
- if `InflationLayersMethod` = 3 then its value defines ratio of total thickness of all layers to the size of underlying surface element,
- if `InflationLayersMethod` = 4 then its value defines ratio of first layer thickness to the size of underlying surface element,
- if `InflationLayersMethod` = 5 then its value defines ratio of last layer element volume to the volume of adjacent tetrahedral element,

Physical unit applies only when `InflationLayersMethod` = 1 or 2 and depends on the geometry input format, in case of IGES it is [mm], in case of STL it is the same physical unit that the geometry was exported in from the geometry pre-processor.

Data type: Float

Possible values: Any > 0 .

Default value: None

InflationLayersGrowthRate

Ratio of thickness of two neighboring inflation layers (geometric progression).

Data type: Float

Possible values: Any ≥ 0 .

Default value: None

All surfaces on which the inflation layers are to be created must currently have identical values of parameters `InflationLayers`, `InflationLayersMethod`, `InflationLayersThickness`, and `InflationLayersGrowthRate`, as this functionality is still under development.

Appendix A Structure of the MSH File

```

$MeshFormat
4.1 0 8
$EndMeshFormat
$PhysicalNames
13
2 1 "V1.p1.1"
2 2 "V1.p2.1"
2 3 "V2.16.1"
2 4 "V1.p4.1"
2 5 "V1.p5.1"
2 6 "V1.p6.1"
2 7 "V2.11.1"
2 8 "V2.12.1"
2 9 "V2.13.1"
2 10 "V2.14.1"
2 11 "V2.15.1"
3 12 "V1"
3 13 "V2"
$EndPhysicalNames
$Entities
...
$EndEntities
$Nodes
...
$EndNodes
$Elements
13 868 1 868
2 1 2 32
1 80 1 13
...
32 75 81 79
...
2 11 2 32
321 170 3 19
...
352 165 171 169
3 1 4 256
353 79 175 116 176
...
608 83 33 91 32
3 2 4 260
609 138 179 180 182
...
868 63 153 139 142
$EndElements

```

\$MeshFormat
 4.1 0 8
\$EndMeshFormat
\$PhysicalNames — Description of groups of geometric entities follows
 13 — Total number of geometric entity groups
 2 1 "V1.p1.1" — Geometric entities of 2nd order (surfaces)
 2 2 "V1.p2.1" — Names of surfaces
 2 3 "V2.16.1"
 2 4 "V1.p4.1"
 2 5 "V1.p5.1"
 2 6 "V1.p6.1"
 2 7 "V2.11.1" — Identification numbers of geometric entities
 2 8 "V2.12.1"
 2 9 "V2.13.1"
 2 10 "V2.14.1"
 2 11 "V2.15.1"
 3 12 "V1" — Geometric entities of 3rd order (volumes)
 3 13 "V2" — Names of volumes
\$EndPhysicalNames
\$Entities — Description of geometric entities follows
 ...
\$EndEntities
\$Nodes — Data of mesh nodes follows
 ...
\$EndNodes
\$Elements — Data of finite elements follows
 13 868 1 868 — Total number of groups of finite elements
 2 1 2 32 — Total number of finite elements
 1 80 1 13 — Range of identification numbers of the finite elements
 ...
 32 75 81 79
 ...
 2 11 2 32 — Geometric entity of 2nd order (surface)
 321 170 3 19 — Identification number of the geometric entity
 ...
 352 165 171 169 — Finite element of type 2 (triangle)
 ...
 3 1 4 256 — The number of finite elements falling under this entity
 353 79 175 116 176 — Identification number of a finite element
 ...
 608 83 33 91 32 — Identification numbers of nodes forming the element
 3 2 4 260 — Geometric entity of 3rd order (volume)
 609 138 179 180 182 — Finite element of type 4 (tetrahedra)
 ...
 868 63 153 139 142
\$EndElements