Mesh generation with Gmsh

Dr. Sayan Adhikari

Department of Physics, University of Oslo, Norway

PTetra Workshop, January 25, 2022

Workshop URL

Contents

Introduction to Gmsh

Gmsh is

- an open source 3D finite element mesh generator
- a CAD engine ▸ Example
- a post-processing tool ▸ Example
- available for different distributions (e.g. Windows, Linux, MacOS)

▸ Learn more about Gmsh

Introduction to Gmsh

Gmsh is

- an open source 3D finite element mesh generator
- a CAD engine ▸ Example
- a post-processing tool ▸ Example
- available for different distributions (e.g. Windows, Linux, MacOS)

Its design goal is to provide a fast, light and user-friendly meshing tool with parametric input and advanced visualization capabilities.

▸ Learn more about Gmsh

A little more about Gmsh

Gmsh can be used

- through a GUI
- using a command line through Gmsh's own scripting language

A little more about Gmsh

Gmsh can be used

- through a GUI
- using a command line through Gmsh's own scripting language

In today's session, we shall learn a method with a combination of both.

A little more about Gmsh

The geometry of any given problem is represented via a file “`<filename>.geo`”. Such files can be created:

- using a text editor like vim.
- through the GUI

A little more about Gmsh

The geometry of any given problem is represented via a file "`<filename>.geo`". Such files can be created:

- using a text editor like vim.
- through the GUI

To open the Gmsh GUI

```
#> gmsh
```

A little more about Gmsh

The geometry of any given problem is represented via a file "`<filename>.geo`". Such files can be created:

- using a text editor like vim.
- through the GUI

To open the Gmsh GUI

```
#> gmsh
```

To open a `.geo` file using the Gmsh GUI

```
#> gmsh <filename>.geo
```

A little more about Gmsh

The geometry of any given problem is represented via a file "`<filename>.geo`". Such files can be created:

- using a text editor like vim.
- through the GUI

To open the Gmsh GUI

```
#> gmsh
```

To open a `.geo` file using the Gmsh GUI

```
#> gmsh <filename>.geo
```

Note: Remember to activate you `conda` environment

```
#> conda activate ptetra
```

A little more about Gmsh

How to write a `.geo` file?

Use vim/vi/nano or any of your favorite text editor to create a `HelloWorld.geo` file.

```
#> vim HelloWorld.geo
```

A little more about Gmsh

How to write a `.geo` file?

Use vim/vi/nano or any of your favorite text editor to create a `HelloWorld.geo` file.

```
#> vim HelloWorld.geo
```

Now, let's make our own first geometry file!

Basic elements of Gmsh

How to define points?

Points are defined as:

```
Point(id) = {x, y, z, cl};
```

`id`: unique id assigned to each element with a specific type

`x, y, z`: coordinates of the specific point

`cl`: mesh element size at this point

Basic elements of Gmsh

How to define points?

Points are defined as:

```
Point(id) = {x, y, z, cl};
```

`id`: unique id assigned to each element with a specific type

`x, y, z`: coordinates of the specific point

`cl`: mesh element size at this point

Example

```
Point(1) = {0, 0, 0, 0.1};
```

Basic elements of Gmsh

How to define points?

Points are defined as:

```
Point(id) = {x, y, z, cl};
```

`id`: unique id assigned to each element with a specific type

`x, y, z`: coordinates of the specific point

`cl`: mesh element size at this point

Example

```
Point(1) = {0, 0, 0, 0.1};
```

Now, let's see it in action!

Basic elements of Gmsh

Example of a .geo file with 8 points

```
Point(1) = {0, 0, 0, 0.1};
Point(2) = {1, 0, 0, 0.1};
Point(3) = {0, 1, 0, 0.1};
Point(4) = {0, 0, 1, 0.1};
Point(5) = {0, 1, 1, 0.1};
Point(6) = {1, 0, 1, 0.1};
Point(7) = {1, 1, 0, 0.1};
Point(8) = {1, 1, 1, 0.1};
```

Basic elements of Gmsh

How to define lines?

Points are defined as:

```
Line(id) = {start, end};
```

`id`: unique id assigned to each element with a specific type

`start, end`: start and end points of a line

Basic elements of Gmsh

How to define lines?

Points are defined as:

```
Line(id) = {start, end};
```

`id`: unique id assigned to each element with a specific type

`start, end`: start and end points of a line

Example

```
Point(1) = {0, 0, 0, 0.1};
Point(2) = {1, 0, 0, 0.1};
Line(1) = {1, 2};
```

Basic elements of Gmsh

How to define lines?

Points are defined as:

```
Line(id) = {start, end};
```

`id`: unique id assigned to each element with a specific type

`start, end`: start and end points of a line

Example

```
Point(1) = {0, 0, 0, 0.1};
Point(2) = {1, 0, 0, 0.1};
Line(1) = {1, 2};
```

Now, let's see it in action!

Basic elements of Gmsh

Example: Eight points connected via lines (Cube)

```
Point(1) = {0, 0, 0, 0.1};
Point(2) = {1, 0, 0, 0.1};
Point(3) = {0, 1, 0, 0.1};
Point(4) = {0, 0, 1, 0.1};
Point(5) = {0, 1, 1, 0.1};
Point(6) = {1, 0, 1, 0.1};
Point(7) = {1, 1, 0, 0.1};
Point(8) = {1, 1, 1, 0.1};
Line(1) = {5, 3};
Line(2) = {3, 7};
Line(3) = {7, 8};
Line(4) = {8, 5};
Line(5) = {4, 1};
Line(6) = {1, 2};
Line(7) = {2, 6};
Line(8) = {6, 4};
Line(9) = {5, 4};
Line(10) = {8, 6};
Line(11) = {7, 2};
Line(12) = {3, 1};
```

Basic elements of Gmsh

How to define circles/circular arc?

Points are defined as:

```
Circle(id) = {start, center, end};
```

`id`: unique id assigned to each element with a specific type

`start, end`: start and end points of an arc

`center`: center of the arc/circle

Basic elements of Gmsh

How to define circles/circular arc?

Points are defined as:

```
Circle(id) = {start, center, end};
```

`id`: unique id assigned to each element with a specific type

`start, end`: start and end points of an arc

`center`: center of the arc/circle

Example

```
Point(1) = {0, 0, 0, 0.1};
Point(2) = {1, 0, 0, 0.1};
Point(3) = {0, 1, 0, 0.1};
Circle(1) = {2, 1, 3};
```

Basic elements of Gmsh

How to define circles/circular arc?

Points are defined as:

```
Circle(id) = {start, center, end};
```

`id`: unique id assigned to each element with a specific type

`start, end`: start and end points of an arc

`center`: center of the arc/circle

Example

```
Point(1) = {0, 0, 0, 0.1};
Point(2) = {1, 0, 0, 0.1};
Point(3) = {0, 1, 0, 0.1};
Circle(1) = {2, 1, 3};
```

Now, let's see it in action!

Setting up a geometry for PTetra

PTetra Workshop Directory

```
PTetraWorkshop
├── BLAS-3.10.0
├── Geometry
│   ├── sphere_0.5R.geo ............................ base .geo file for sphere case
│   ├── cylinder_0.5R_5L.geo ...................... base .geo file for cylinder case
│   └── msh2topo.dat
├── PTetra.zip
├── PTetra
├── README.md
├── environment.yml
├── funcs.py
└── plot.py
```

Setting up a geometry for PTetra

Setting up geometry parameters

Open `sphere_0.5R.geo` using any text editor.

```
#>vim sphere_0.5R.geo
```

Setting up a geometry for PTetra

Setting up geometry parameters

Open `sphere_0.5R.geo` using any text editor.

```
#>vim sphere_0.5R.geo
```

```
// STEP 1:  SET VARIABLES
debye = 0.00690; // Electron debye length for n=1e11 and T=1000
r = 0.5*debye; // Inner radius
R = TBD; // Outer radius
Res = TBD; // Resolution on outer boundary
res = TBD; // Resolution on inner boundary
```

Setting up a geometry for PTetra

Deciding the system length and local grid resolution

```
// STEP 1:  SET VARIABLES
debye = 0.00690; // Electron debye length for n=1e11 and T=1000
r = 0.5*debye; // Inner radius
R = TBD; // Outer radius How to decide such a number??
Res = TBD; // Resolution on outer boundary
res = TBD; // Resolution on inner boundary
```

Answer: There is no simple rule. But, the best practice is to consider the system length so big that ensures the E-field due to the object is zero at the boundary (Dirichlet Boundary).

Setting up a geometry for PTetra

Deciding the system length and local grid resolution

```
// STEP 1:  SET VARIABLES
debye = 0.00690; // Electron debye length for n=1e11 and T=1000
r = 0.5*debye; // Inner radius
R = r+10*debye; // Outer radius How to decide such a number??
Res = TBD; // Resolution on outer boundary
res = TBD; // Resolution on inner boundary
```

Answer: There is no simple rule. But, the best practice is to consider the system length so big that ensures the E-field due to the object is zero at the boundary (Dirichlet Boundary).

Setting up a geometry for PTetra

Deciding the system length and local grid resolution

```
// STEP 1:  SET VARIABLES
debye = 0.00690; // Electron debye length for n=1e11 and T=1000
r = 0.5*debye; // Inner radius
R = r+10*debye; // Outer radius
What about the local grid resolution??
Res = TBD; // Resolution on outer boundary
res = TBD; // Resolution on inner boundary
```

Answer: The well accepted criterion for PIC simulations in Cartesian meshes: $\Delta x <\sim 3\lambda_D$. For unstructured mesh, Δx becomes the cell diameter (the largest edge-length of a tetrahedron). The central idea of such is to avoid finite grid instabilities in PIC simulations.

Setting up a geometry for PTetra

Deciding the system length and local grid resolution

```
// STEP 1:  SET VARIABLES
debye = 0.00690; // Electron debye length for n=1e11 and T=1000
r = 0.5*debye; // Inner radius
R = r+10*debye; // Outer radius
What about the local grid resolution??
Res = 1.5*debye; // Resolution on outer boundary
res = r/5; // Resolution on inner boundary
```

Answer: The well accepted criterion for PIC simulations in Cartesian meshes: $\Delta x <\sim 3\lambda_D$. For unstructured mesh, Δx becomes the cell diameter (the largest edge-length of a tetrahedron). The central idea of such is to avoid finite grid instabilities in PIC simulations.

Setting up a geometry for PTetra

Building geometry Gmsh GUI

```
#>gmsh sphere_0.5R.geo
```

Setting up a geometry for PTetra

Building geometry Gmsh GUI

```
#>gmsh sphere_0.5R.geo
```

Now, let's move to the live session

Setting up a geometry for PTetra

Few useful things of **Gmsh** GUI

Made a mistake?? Go to the left panel

```
Modules
 ├── Geometry
 │    ├── Elementary entities
 │    ├── Physical groups
 │    ├── Reload script
 │    ├── Remove last script command
 │    └── Edit script ........................
 ├── Mesh
 └── Solver
```

It will open up a graphical text editor. Delete the last command and save it.

Setting up a geometry for PTetra

Few useful things of **Gmsh** GUI

Made a mistake?? Go to the left panel

```
Modules
├─ Geometry
│   ├─ Elementary entities
│   ├─ Physical groups
│   ├─ Reload script
│   ├─ Remove last script command
│   └─ Edit script
├─ Mesh
└─ Solver
```

Edit script .... It will open up a graphical text editor. Delete the last command and save it.

Setting up a geometry for PTetra

Few useful things of **Gmsh** GUI

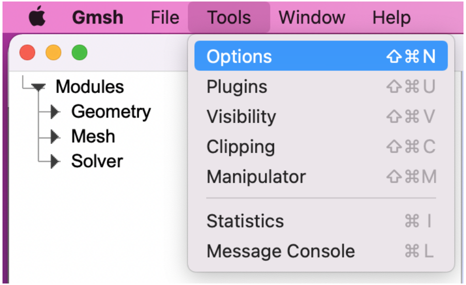
Made a mistake?? Go to the left panel

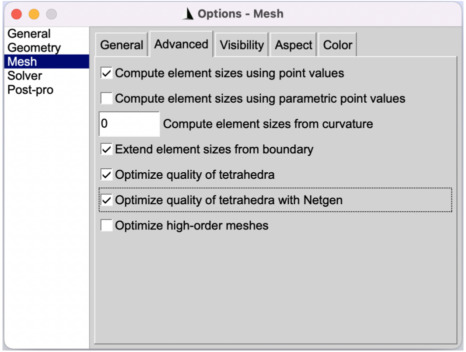
```
Modules
├── Geometry
│   ├── Elementary entities
│   ├── Physical groups
│   ├── Reload script
│   ├── Remove last script command
│   └── Edit script ........................
├── Mesh
└── Solver
```

It will open up a graphical text editor. Delete the last command and save it.

To export/save the Mesh using **Gmsh** GUI

Go to the **`Tools`** menu on the top left corner and click on **`Options`**. Then click on Mesh on the left and open **`Advanced`** tab. Check the box named `Optimize quality of tetrahedra with Netgen`. Remember to click on `Save Options as Defaults` before closing **Gmsh**.





Mesh generation for PTetra

Steps to generate Mesh using **Gmsh** GUI

Go to the left panel and click on `Mesh`

```
Modules
 ├── Geometry
 ├── Mesh
 │    ├── Define
 │    ├── 1D
 │    ├── 2D ........................................ To generate 2D Mesh.
 │    ├── 3D ........................................ To generate 3D Mesh.
 │    ├── ...
 │    ├── ...
 │    ├── Inspect
 │    └── Save
 └── Solver
```

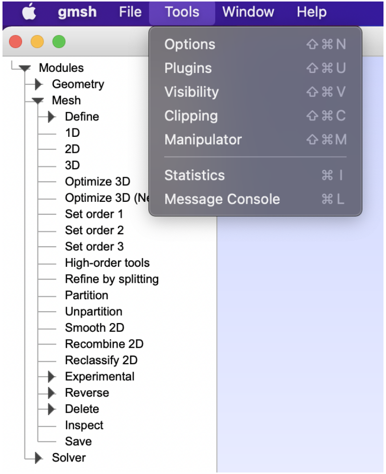
Mesh generation for PTetra

Steps to generate Mesh using **Gmsh** GUI

Go to the left panel and click on `Mesh`

```
Modules
 ├─ Geometry
 ├─ Mesh
 │   ├─ Define
 │   ├─ 1D
 │   ├─ 2D ........................................ To generate 2D Mesh.
 │   ├─ 3D ........................................ To generate 3D Mesh.
 │   ├─ ...
 │   ├─ ...
 │   ├─ Inspect
 │   └─ Save
 └─ Solver
```

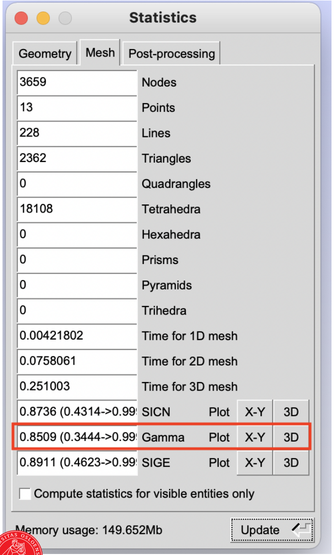
Mesh generation for PTetra



Understanding the quality of Meshes

Go to the `Tools` menu on the top left corner and click on `Statistics`.

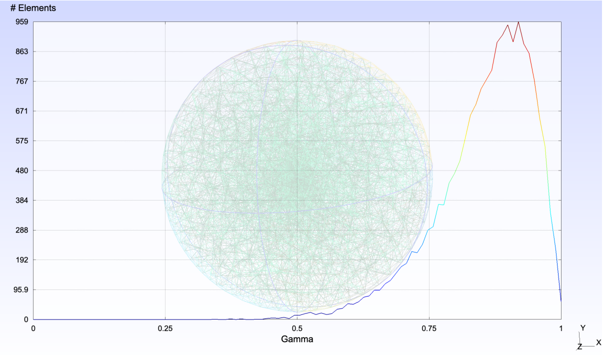
Mesh generation for PTetra



Understanding the quality of Meshes

Go to the `Tools` menu on the top left corner and click on `Statistics`.

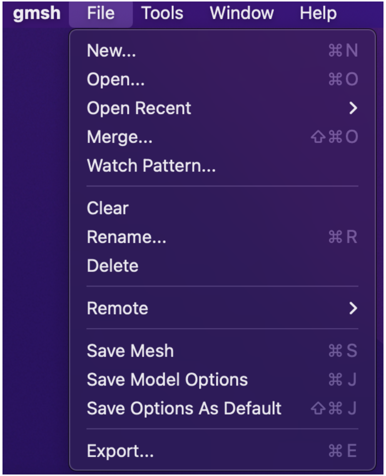
Mesh generation for PTetra



Understanding the quality of Meshes

Go to the `Tools` menu on the top left corner and click on `Statistics`.

Mesh generation for PTetra



To export/save the Mesh using Gmsh GUI

Go to the File menu on the top left corner and click on export. The default save as option should be "`Guess from Extension(*.*)`." Use "`.msh`" extension and choose `Version 2 ASCII` when prompts to save.

Mesh generation for PTetra

Edit `msh2topo.dat`

#>**vim** msh2topo.dat

Replace the `.msh` filename.

```
$begin
nfields=0
sphere.msh
$end
```

Run `msh2topo`

#>**./msh2topo**

Rename `msh2topo.out`

#>**mv** msh2topo.out object.topo

Mesh generation for PTetra

Edit `msh2topo.dat`

```
#>vim msh2topo.dat
```

Replace the `.msh` filename.

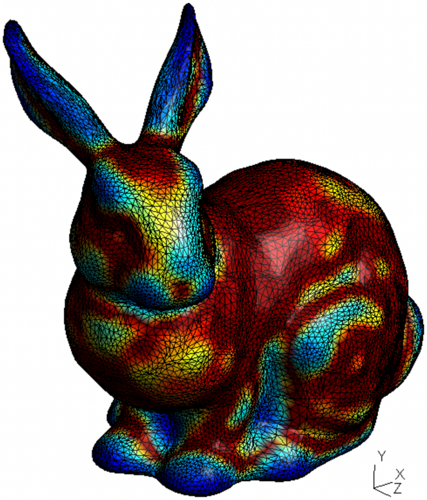
```
$begin
nfields=0
sphere.msh
$end
```

Run `msh2topo`

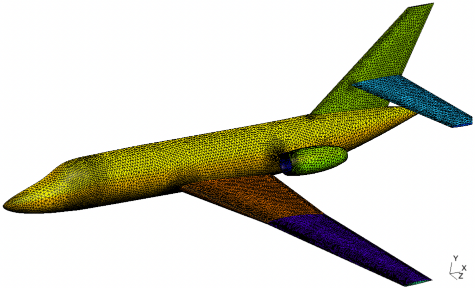
```
#>./msh2topo
```

Rename `msh2topo.out`

```
#>mv msh2topo.out object.topo
```



Stanford Bunny



Dassault Falcon Aircraft

Thank you