Gmsh Workshop
Advanced Mesh Generation Techniques

23-24 May 2013

1 Meshing Tubular Geometries

Starting from a triangulation (in STL format) obtained from image segmentation, we will learn how to create a high quality volume mesh for tubular geometries such as blood vessels or airways. This new fully automatic procedure is implemented in Gmsh and relies on a centerline description of the geometry.

The centerline field is very useful as it enables to generate meshes with mesh sizes that are function of the vessel radius and with mesh metrics that are aligned with the directions the centerlines. Moreover, it enables to cut the initial triangulation if smaller tubes that be can subsequently easily remeshed with our 2D mesh quadrangular and triangular mesh algorithms. The Centerline field of gmsh takes as input the centerlines (quite complex to compute) that are computed using the open-source software vmtk. There are three possible actions for the Centerline Field (closeVolume, extrudeWall, reMesh):

```
Field[1] = Centerline;
Field[1].FileName = "centerlines.vtk";
Field[1].nbPoints = 25; //number of mesh elements in a circle

//Close in and outlets with planar faces
Field[1].closeVolume =1;

//Extrude in the outward direction a vessel wall
Field[1].extrudeWall =1;
Field[1].nbElemLayer = 4; //number of layers
Field[1].hLayer = 0.2;// extrusion thickness given as percent of vessel radius

//Remesh the initial stl
Field[1].reMesh =1;

Field[1].run;
Background Field = 1;
```

Suppose you have a tubular geometry. For this example we will consider cerebral blood vessels with an aneurysm. With this geometry, we would like to perform 3D (possibly FSI) blood flow simulation. We only have an image segmentation of the lumen of the vessel but not of the vessel thickness. We will see how to create a 3D mesh with an extruded wall that has a thickness that is function of the vessel radius.

Perform the following different steps to learn how to use the centerline field:

- Start from the STL file that is a triangulation of a cerebral aneurysm.
  Open the file: `gmsh aneurysm.stl`

- From this STL you can create the centerline of this tubular geometry (in VTK format) using the vmtk software (`http://www.vmtk.org/Tutorials/Centerlines`)
  (i) First extend the stl file, i.e extend the inlet and outlets so that the computed centerlines extend out of the inlet and outlets of the input geometry (you can also simple run the script `./myExtend.sh aneurysm.stl`):
  ```
  vmtk vmtksurfacereader -ifile aneurysm.stl
    --pipe vmtkcenterlines -seedselector openprofiles
    --pipe vmtkflowextensions -adaptivelength 1 -extensionratio 1.5
    -interactive 0 --pipe vmtksurfacewriter -ofile extended.vtk
  ```
  The vmtk script opens a graphical interface which you can quit by pressing 'q'. Then you have to select the inlet (type '`1`') and for the outlets type '`enter`' which means that you consider all the remaining boundaries to be outlets.
  (ii) Then compute the centerlines of the extended tubular structure (you can also simple run the script `./myCenterlines.sh extended.vtk`) :
  ```
   vmtk vmtkcenterlines -seedselector openprofiles
       -ifile extended.vtk  -ofile centerlines.vtp
       --pipe vmtksurfacewriter -ifile centerlines.vtp -ofile centerlinesANEU.vtk
  ```
  You can run those commands on the virtual machine of the workshop using:
  `scp aneurysm.stl vm@amzyan.local:.` to copy the file,
  `ssh -X vm@amzyan.local` to connect to the virtual machine and
  `scp vm@amzyan.local:centerlinesANEU.vtk .` to copy it back.
  (iii) Open the created centerlines with gmsh `gmsh centerlinesANEU.vtk`. In order to see the lines of the centerlines you have to select in Gmsh the Tools > Options > Mesh >Llines.
  If you have some problems with this step, you will find the created centerline in the geom repository.

- Have a look with 'vi' or 'textedit' at `aneurysm_centerlines.geo` in order to understand how to define the centerline field. As a first example we have chosen not to remesh the surface but only to close the volume (see the centerline field action `Field[1].closeVolume =1;`) and to mesh the volume. You can see in the geo file that we have selected the MeshAdapt algorithm for the 2D meshing of the inlet and outlet planar faces as well as Tetgen for the 3D volume mesher. Run `gmsh aneurysm_centerlines.geo -3` and have a look at the volume mesh and look at the physical tags (Tools >Visibility>Physical Entities) that are nice for a CFD computation.

- Suppose now the quality of your triangulation is too low to create a volume mesh or that you wish to have a quad surface mesh a structured layer for the vessel wall.
  Edit the `aneurysm_centerlines.geo` file and uncomment the two following actions:
  `Field[1].extrudeWall =1;` and `Field[1].reMesh =1;`
  Change also the 2D surface mesh (choosing delquad or `Mesh.algorithm= 8`), and select a recombination of the triangles `Mesh.RecombineAll = 1;`
  At first open the geo file with gmsh: `gmsh aneurysm_centerlines.geo`. This may take a while because the automatic algorithm will split your tubular mesh into many small tubes. Have a look (Tools > Visibility) at all the split surfaces.
  Now run : `gmsh aneurysm_centerlines.geo -3` and press `1[Continue]` when asking if you want a mixed mesh. Have a look at the resulting mesh containing tetrahedra, hexahedra and pyramids. Have a closer look by using the Clipping Tool of Gmsh as well as coloring the mesh elements by element type.

- Additionally you can also create an isotropic mesh by choosing the bamg surface mesher (2D) with the mmg3D volume mesher. Comment the ExtrudeWall action and the recombineAll action and run `gmsh aneurysm_centerlines.geo -3`

- Other examples are in the geom repository (aorta and carotid bifurcation)

You can find some more information on the algorithm in the following paper :

E.Marchandise, C. Geuzaine, J.F. Remacle. Cardiovascular and lung mesh generation based on centerlines, international journal for numerical methods in biomedical engineering, online 2013.

2 Quad and hex meshes

The aim here is to generate a hex-domiannt mesh on a complex geometry. Two geometries are available: `BLADES.step` and and `submarine.stp`. Note that the first geometry will be way simpler to deal with than the second one. Importants things to know:

- Create a `.geo` file and use the `Merge` command to import step files.
- Use 2D and 3D algorithms that allow to generate triangles and tets that will favorize quad/hex merging. Use `Mesh.Algorithm3D = 9;` and `Mesh.Algorithm = 8;`
- Do not allow mesh smoothing (not yet generalized to non equilateral elements). Use `Mesh.Smoothing = 0;`
- When a surface has an ugly parametrization, reparametrize it using compounds.
- Ask for generating hex dominant meshes: `Mesh.Recombine3DAll = 1;`

3 Boundary Layer and High-Order Meshing

We are going to create a high-order hybrid mesh for the case of a high-lift airfoil. The geometry of the airfoil is defined by 7 splines (2 for the slat, 2 for the main wing and 3 for the flap), supported by a large number of points. This geometry is contained in the script file "airfoil_geometry.geo".

Our mesh will be defined in a separate script file for the sake of clarity. Do not forget to reload the file (*Modules* → *Geometry* → *Reload*) each time you add a command!

3.1 Creating the domain

- Start with a blank script file : `gmsh high-lift.geo` in the command shell
- We need to merge the geometry defined in the file "airfoil_geometry.geo". However, we need to define before the value of a variable `lcWall` that is used in "airfoil_geometry.geo" to assign an element size to each point:

  ```
  lcWall = 100;
  Merge "airfoil_geometry.geo";
  ```

- Now we want to create the circular outer boundary of the domain. We will first define variables that represent the x-position of the domain center and its radius, as well as the element size on that boundary:

  ```
  xC = 400;
  RFar = 5000;
  lcFar = 1000;
  ```

  And we can define the point at the center of the domain:

  ```
  Point(2000) = {xC, 0, 0, lcFar};
  ```

  In the same manner, you can add points 2001, 2002 and 2003 that are located respectively at $(\mathrm{xC} + \mathrm{RFar}, 0, 0)$, $(\mathrm{xC} + \mathrm{RFar} \cdot \cos(\pi/3), \mathrm{RFar} \cdot \sin(\pi/3), 0)$ and $(\mathrm{xC} + \mathrm{RFar} \cdot \cos(-2\pi/3), \mathrm{RFar} \cdot \sin(-2\pi/3), 0)$. Tip: don't forget the capital letter in commands `Cos`, `Sin` and `Pi`!

- We can now define the first arc of the circular outer boundary:

```
Circle(10) = {2001, 2000, 2002};
```

You can create the other two arcs (number 11 and 12) in the same way.

- Now we need to define the "line loops" that will bound the surface to be meshed. Displaying the labels identifying the lines of the geometry will help: (*Tools* → *Options* → *Geometry* → *Visibility* → *Line labels*). Let's start with the slat:

  ```
  Line Loop(13) = {1, 2};
  ```

  Create line loops 14, 15 and 16 for the main wing, the flap and the outer boundary respectively.

- We can finally define the surface to be meshed as line loop 16 with "holes" defined by line loops 13, 14 and 15:

  ```
  Plane Surface(17) = {16, 13, 14, 15};
  ```

3.2 Boundary Layer Meshing

- At this point, we can obtain a first mesh (*Modules* → *Mesh* → *2D*).

- We can play with the values of `lcWall` and `lcFar` to obtain a suitable mesh density. The mesh can be refined in the curved parts of the geometry with the command:

  ```
  Mesh.CharacteristicLengthFromCurvature = 1;
  ```

- We will now add a boundary layer mesh by defining an element size field and setting the boundary layer mesher to use it:

  ```
  Field[1] = BoundaryLayer;
  Field[1].EdgesList = {1, 2, 3, 4, 5, 6, 7};
  Field[1].hfar = lcFar;
  Field[1].hwall_n = 0.1;
  Field[1].hwall_t = 100;
  Field[1].ratio = 1.4;
  Field[1].thickness = 5;
  BoundaryLayer Field = 1;
  ```

  The boundary layer mesh is made up of triangles, but you can turn them into quadrangles by adding:

  ```
  Field[1].Quads = 1;
  ```

- There is a good chance that the boundary layer mesh does not correspond to your requirements where the angle between two successive boundary edges is too high. You can then adjust the fan angle:

  ```
  Field[1].fan_angle = 50;
  ```

  The creation of fan at protruding points of the geometry can be controlled by:

  ```
  Field[1].NodesList = {1, 2, 9, 5, 6};
  ```

3.3 High-Order Meshing

- We will now try to convert our hybrid mesh into a quadratic mesh: *Modules* → *Mesh* → *Set order 2*.
- Zoom on the curved parts of the geometry: do you see any problem with tangled elements? In order to check the validity of your mesh, you have two options at hand:
  - The statistics window (*Tools* → *Statistics* → *Update*). The "Disto" box shows you the range of minimal scaled Jacobian over all elements of the mesh. If the minimum is negative, one or more elements may be invalid.
  - The "AnalyseCurvedMesh" plug-in (*Tools* → *Plugins* → *AnalyseCurvedMesh*). You can check the different possibilities in the "Help" tab. If you decide to hide all the valid elements to better visualize the invalid ones, you can reset the visualization with the visibility tool (*Tools* → *Visibility*).

  The background theory about the validity of curvilinear meshes, as well as the related algorithms implemented in Gmsh, can be found in the following paper: A. Johnen, J.-F. Remacle and C. Geuzaine, Geometrical Validity of Curvilinear Finite Elements, JCP 233:359–372.
- We will now try to repair the mesh that contains invalid elements with the optimisation tool (*Modules* → *Mesh* → *Optimize high order*). At first, you can set a Jacobian range of (0.5-10). You may want to increase the number of layers, so that large blobs allow the elements to deform enough to reach the target minimum Jacobian. The other parameters can remain untouched. After running the tool (*Apply* at the bottom of the optimization window), you can check the mesh validity again.
- Look at the concave parts of the geometry. Are you satisfied with the shape of the elements there? For the curvature of the boundary to "propagate" into the domain, we can constrain the maximum Jacobian: return to the optimization window, change the Jacobian range to (0.5-2) and run the optimization tool again. You should be able to see the improved mesh quality on the concave part of the geometry.

4 Bonus: High-Order Meshing and Python Interface

This part of the workshop requires the installation of the binary Gmsh libraries:

- `http://geuz.org/gmsh/bin/Linux/gmsh-svn-Linux64-dynamic.tgz` (Linux 64bit)
- `http://geuz.org/gmsh/bin/MacOSX/gmsh-svn-MacOSX-dynamic.tgz` (Mac)

Then, the Python interface can be activated by issuing the following commands in the gmshpy subdirectory:

```
python setup.py build
sudo python setup.py install
```

Please bear in mind that the gmsh dynamic library (from the lib directory) should be located in a directory listed in your `(DY)LD_LIBRARY_PATH`.

The instructions for this part will be provided orally.