TP2: Mesh adaptation with the Mmg platform

Algiane Froehly

13 décembre 2017

1 Goals :

- Adapt a mesh to a given size map ;
- Compute an isotropic/anisotropic size map based on the interpolation error.

2 Mmg platform in short

The Mmg platform gathers software dedicated to simplicial mesh modifications :

- mmg2d : for 2D meshes ;
- mmgs : for 3D surface meshes ;
- mmg3d : for 3D volume meshes.

All this software allow quality improvement, mesh adaptation to a size map (isotropic/anisotropic) and level-set discretization.

Additional documentation can be founded here :
`http://www.mmgtools.org/mmg-remesher-try-mmg/mmg-remesher-tutorials`
and here :
`http://www.mmgtools.org/mmg-remesher-try-mmg/mmg-remesher-options`

2.1 Installation

1. Clone the Mmg repository and build the applications and libraries :

```
$ git clone https://github.com/MmgTools/mmg.git
$ cd mmg
$ mkdir build
$ cd build
$ cmake ..
$ make
```

2. Add the path of the `build/bin` folder to you `PATH` variable to be able to run the applications from your terminal without adding the full binary path (in the command line, `$PATH_TO_BIN` must be replaced by your path through the Mmg repository :

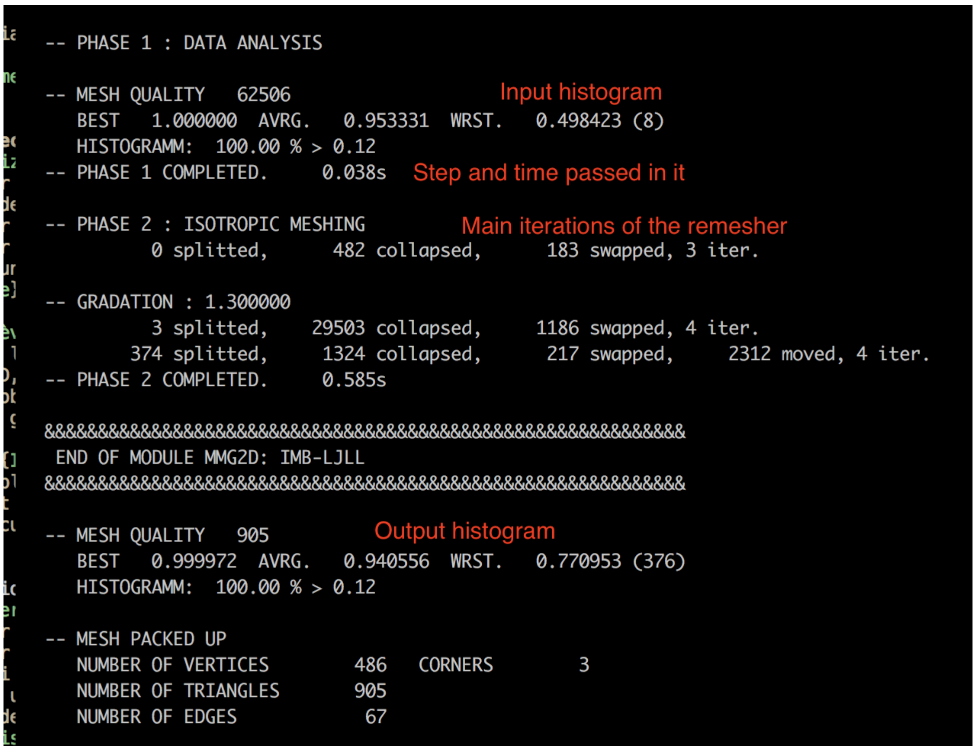
```
$ echo "PATH=$PATH_TO_BIN:$PATH" >> ~/.bashrc
$ source ~/.bashrc
```

3 A first run of the remesher

To run the remesher, you must give the application name followed by the path and mesh name (the naca mesh of the previous TP is provided in the `TP/Data` directory of this repository) :

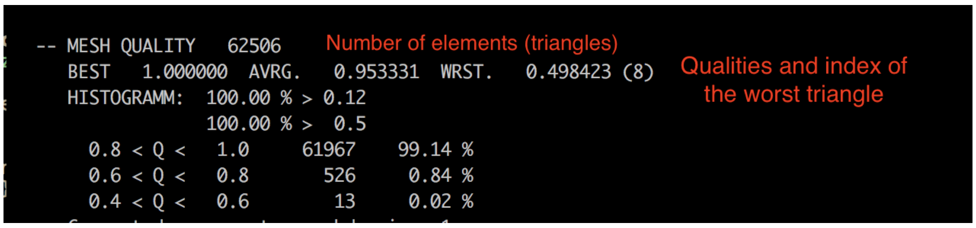
```
$ cd TP
$ mmg2d_O3 Data/naca_embedded.mesh
```

By default, Mmg creates a mesh in the same path and of the same extension than the input mesh (`.mesh` here) with the `.o` prefix before the extension, so here : `Data/naca_embedded.o.mesh`.



```
  -- PHASE 1 : DATA ANALYSIS

  -- MESH QUALITY   62506
     BEST   1.000000  AVRG.   0.953331  WRST.   0.498423 (8)
     HISTOGRAMM:  100.00 % > 0.12
  -- PHASE 1 COMPLETED.     0.038s

  -- PHASE 2 : ISOTROPIC MESHING
            0 splitted,      482 collapsed,      183 swapped, 3 iter.

  -- GRADATION : 1.300000
            3 splitted,    29503 collapsed,     1186 swapped, 4 iter.
          374 splitted,     1324 collapsed,      217 swapped,     2312 moved, 4 iter.
  -- PHASE 2 COMPLETED.     0.585s

  &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
   END OF MODULE MMG2D: IMB-LJLL
  &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

  -- MESH QUALITY   905
     BEST   0.999972  AVRG.   0.940556  WRST.   0.770953 (376)
     HISTOGRAMM:  100.00 % > 0.12

  -- MESH PACKED UP
     NUMBER OF VERTICES        486   CORNERS        3
     NUMBER OF TRIANGLES       905
     NUMBER OF EDGES            67
```

FIGURE 1 – Default Mmg output.


Number of elements (triangles)
Qualities and index of the worst triangle

```
  -- MESH QUALITY   62506
     BEST   1.000000  AVRG.   0.953331  WRST.   0.498423 (8)
     HISTOGRAMM:  100.00 % > 0.12
                  100.00 % >  0.5
       0.8 < Q <   1.0      61967    99.14 %
       0.6 < Q <   0.8        526     0.84 %
       0.4 < Q <   0.6         13     0.02 %
```

FIGURE 2 – Detailed quality histogram.

3.1 The Mmg output

You can see the Mmg output in your terminal. By default, Mmg prints (see figure 1) :

- The different phases of the algorithm (analysis step, remeshing step...) and the time spent in each of this steps ;
- some info about the input/output qualities histogram ;
- the final mesh statistics (number of nodes, elements and edges).

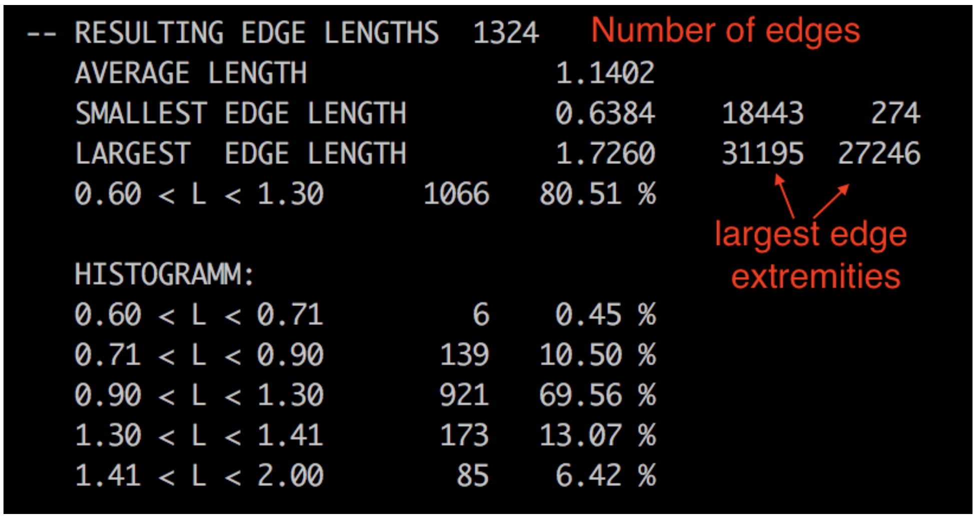
You can change the default verbosity of Mmg with the `-v` option. By default, the verbosity value is setted to 1. If you set the verbosity to 5, `mmg2d_O3 Data/naca_embedded.mesh -v 5`, you will obtain :

- detailed quality histograms (see figure 2) ;
- detailed remeshing steps ;
- edge length histogram (see figure 3).

3.2 Mesh improvement with edge length preservation : -optim option

Open your output mesh in Gmsh : by default, Mmg tries to create a mesh that respect the asked boundary approximation (`-hausd` option, 0.1 by default), the maximal ratio between two adjacent edges (`-hgrad` option, 1.3 by default) and that contains the smallest possible number of points.
If you want to preserve the edge length of the input mesh, you can run Mmg with the `-optim` option :

```
$ mmg2d_O3 Data/naca_embedded.mesh -optim -v 5
```



```
-- RESULTING EDGE LENGTHS  1324
   AVERAGE LENGTH                 1.1402
   SMALLEST EDGE LENGTH           0.6384    18443    274
   LARGEST  EDGE LENGTH           1.7260    31195  27246
   0.60 < L < 1.30      1066     80.51 %

   HISTOGRAMM:
   0.60 < L < 0.71         6      0.45 %
   0.71 < L < 0.90       139     10.50 %
   0.90 < L < 1.30       921     69.56 %
   1.30 < L < 1.41       173     13.07 %
   1.41 < L < 2.00        85      6.42 %
```

FIGURE 3 – Edge length histogram.

Open your mesh in Gmsh and check that the edge lengths are preserved. You can compare the input and output quality histograms.

If you want, you can play with other options : try for example to disable the gradation (`-hgrad -1`) or to ask for a better boundary approximation (`-hmin 0.000001 -hausd 0.0001`).

Why do I need to specify `hmin` in addition to the hausdorff parameter ?

To avoid numerical errors (division by 0) and users mistakes (0 length edges asked), Mmg automatically computes a minimal edge size. If a size map is provided, this minimal edge size is smallest than the smallest asked length but if the user doesn't provide a size map, we must extract a length information from the intial mesh : in this case, by default, Mmg set `hmin` to 0.1 times the mesh bounding box size. In our case, because the naca is a very small object in an infinite box, the default `hmin` value became too large when we ask for a finer boundary approximation.

4 Mesh adaptation to a size map

You can provide to Mmg a size map in a `.sol` file. This file lists, for each node, the prescribed edge length. In isotropic case, the file contains 1 scalar data per node (s). In anisotropic case, it contains a metric tensor :

$$M = R\,\Lambda\,R^T$$

With :

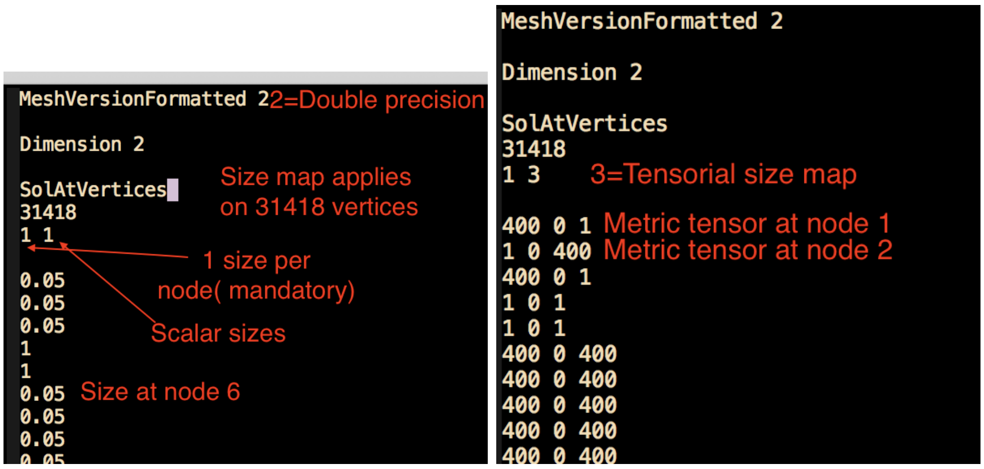
- $\Lambda = (\lambda_j)_j$ a diagonal matrix such as $\lambda_j = 1/s_j^2$ with s the wanted edge length.
- $R = (r_{ij})_{ij}$, an orthonormal matrix such as the r_j vectors gives the direction in which we want the s_j length.

See figure 4 to see an explanation of the .sol file format for an isotropic and an anisotropic size map.

You will find in the `Data` directory two size maps, `naca_iso.sol` and `naca_aniso.sol`. Try to adapt your mesh to each map. For example, for the isotropic map :

```
$ mmg2d_O3 Data/naca_embedded.mesh -sol naca_iso.sol -hausd 0.001 -v 5
```

Again, you can play with the gradation parameter.



```
MeshVersionFormatted 2

Dimension 2

SolAtVertices
31418
1 1

0.05
0.05
0.05
1
1
0.05
0.05
0.05
0.05
```

```
MeshVersionFormatted 2

Dimension 2

SolAtVertices
31418
1 3

400 0 1
1 0 400
400 0 1
1 0 1
1 0 1
400 0 400
400 0 400
400 0 400
400 0 400
400 0 400
```

FIGURE 4 – Isotropic (left) and anisotropic (right) size maps at `Medit` file format.

5 Size map computation to control the error of interpolation of an analytic function over the mesh

5.1 Computation of the nodal values of a 2D analytic function

1. Choose a function, for example, a Gaussian function : $f(x,y) = 10 * exp(-x * x/30 - y * y/30)$ or a sinus one $f(x,y) = 5 * sin(x)$;
2. Compute its nodal values on the mesh nodes. You can start from the `Data/createSol.c` file and fill the `f` function ;
3. Build the application (the `$MMG_PATH` variable must be replaced by your path through the Mmg directory) :

```
$ gcc createSol.c -L $MMG_PATH/build/lib/ -lmmg2d
  -I $MMG_PATH/build/include/ -lm
```

4. This application takes 3 arguments : your inital mesh, an output mesh name at Gmsh format (.msh extension) that will be used to vizualize your solution and the output solution name with `.sol` extension. This solution is created by the application and contains the nodal values of the function :

```
 $ ./a.out naca_embedded.mesh vizu.msh naca_embedded.sol
```

5. you can open `vizu.msh` in Gmsh to check your solution.

5.2 Computation of an anisotropic size map to control the interpolation error over the mesh

1. Compute the Hessian of the previous analytical function (inside the `hessian` function of the createSol.c file) ;
2. compute $M_V = \frac{2}{9\epsilon} H_u(V)$ with V a mesh node. Be careful : you cannot ask for edges of null length (use a minimal edge length to truncate the 0 eigenvalues of M_V). (You can implement it in the `tensor_size` function of the ttbcreateSol.c file). The metric tensor is symetric definite positive, thus, Mmg only stores 3 of the 4 tensor data inside a 1D array : m_{11}, m_{12}, m_{22} (in this order).
3. you can try to vizualize your anisotropic metric field inside Gmsh but note that when drawing the ellipse associated to the tensor, Gmsh plot the eigenvalue (so the size inverse) associated to the eigenvectors ;
4. adapt your mesh and check the results.

5.3 Computation of an isotropic size map to control the interpolation error over the mesh

1. Compute the eigenvalues of M_V in the `eigenvals` function ;
2. compute the maximal eigenvalue λ. Again, this value must be truncated by a minimal size if it is 0 ;
3. Find λ, the maximal eigenvalue and compute $s_v = \frac{1}{\sqrt{\lambda}}$. You can implement this step in the `scalar_size` function of the `createSol.c` file
4. save s_v in a `.sol file` ;
5. adapt your mesh and check the results.