TP2: Mesh adaptation with the Mmg platform

Algiane Froehly

16 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`.

FIGURE 1 – Default Mmg output.

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    Number of edges
   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 %
                                               largest edge
   HISTOGRAMM:                                  extremities
   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 %
```
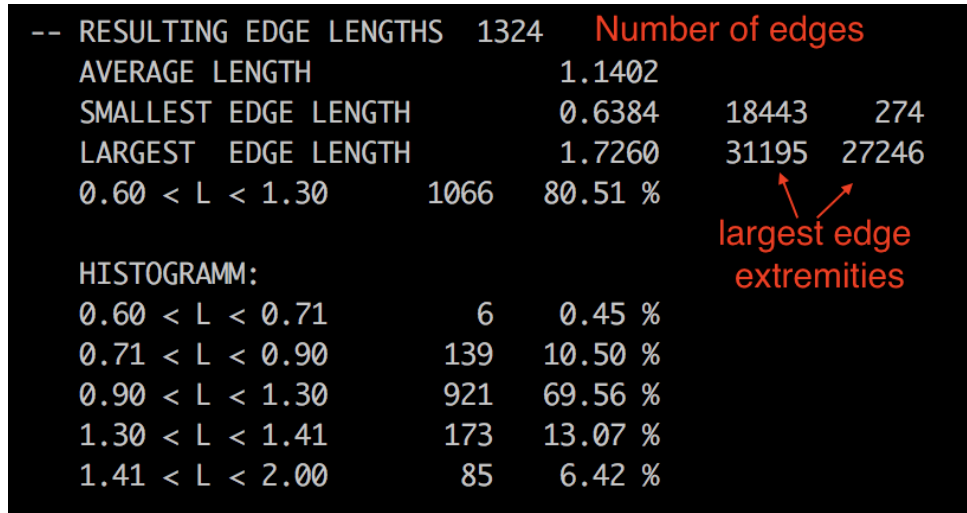
FIGURE 3 – Edge length histogram.

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. In anisotropic case, it contains a metric tensor M that can be diagonalized in the basis of the eigenvectors :

$$M = R \Lambda R^{-1}$$

With :

- $R = (r_{ij})_{ij}$, the matrix of the eigenvectors ;
- $\Lambda = (\lambda_j)_j$ the diagonal matrix of the eigenvalues.

A given eigenvalue and the associated eigenvector are related to the wanted edge length : $\lambda_j = \frac{1}{s_j^2}$ with s_j the wanted length in the direction r_j.

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
```

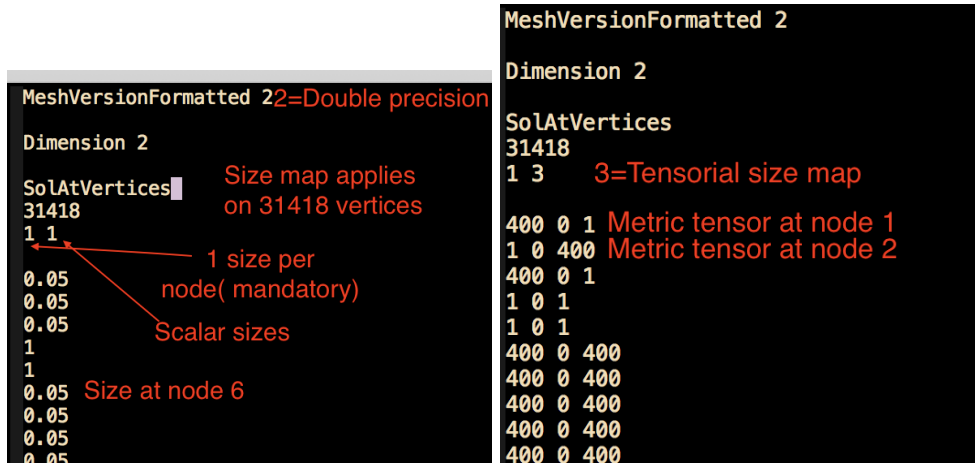Again, you can play with the gradation parameter.

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 $f(x,y) = sin(x/2 + y/2)$;

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, the wanted maximal error of interpolation (ϵ) and the type of metric that you want to compute : 0 for a scalar metric, 1 for a tensorial one. For example, to create the anisotropic metric that prescribes edge lengths allowing to have a maximal error of 0.01 over the `naca_embedded.mesh` file :

```
$ ./a.out naca_embedded.mesh 0.01 1
```

This command will creates 3 files :

- `vizuSolution.msh` that allows to vizualize the analytic function in Gmsh ;
- `vizuMet.msh` that allows to vizualize the computed metric in Gmsh ;
- `metric.sol` to use to adapt your mesh with Mmg.

Note that at this step, the metric computation is not yet implemented thus `vizuMet.msh` and `metric.sol` contain uninitialized values.

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

We remind that, at each mesh node V, we want to compute the matrix $M(V)$ such as :

$$M(V) = \frac{2}{9\epsilon}\,|H_u(V)| = \frac{2}{9\epsilon}\,R\,|\Lambda|\,R^{-1}.$$

5.2.1 Anisotropic size map

You can compute the tensor metric $M(V)$ inside the `tensor_size` function of the `createSol.c` file. Use the `siz` array (of size 3) to store m_{11}, m_{12} and m_{22} ($m_{21} = m_{12}$ so it is useless to store it).

For this :

1. Compute $H(V)$, the Hessian of the previous analytical function at a node V. This matrix is symmetric definite positive, thus, it is possible to store only 3 of the 4 tensor data inside a 1D array : h_{11}, h_{12}, h_{22}. (you can implement this inside the `hessian` function of the `createSol.c` file) ;
2. compute $\bar{H}(V) = \frac{2}{9\epsilon} H(V)$;
3. compute the eigenvectors and the absolute value of the eigenvalues of $\bar{H}(V)$ (you can use the given `eigenvals` function that computes the eigenvectors and eigenvalues of a symetric matrix) ;
4. a null eigenvalue (which physically means that we want an infinite edge) will create numerical issues (division by 0), thus, we need to truncate the maximal edge length. Truncate the maximal edge length by a suitable value (for example, 10. is a suitable value for the `naca_embedded.mesh` mesh).
5. compute $M(V) = R\bar{\Lambda}R^{-1}$, with $\bar{\Lambda}$ the diagonal matrix of the truncated absolute values of the eigenvalues of $\bar{H}(V)$.

Open the `vizuMet.msh` file to vizualize your anisotropic metric field inside Gmsh. Be careful, when drawing the ellipse associated to the tensor, Gmsh plot the eigenvalue (so the size inverse) associated to the eigenvectors.
Last, adapt your mesh using the metric saved in the `metric.sol` file.

5.2.2 Isotropic size map

You can implement the computation of the isotropic edge length at a nod V in the `scalar_size` function of the `createSol.c` file :

1. Perform the 4 steps of the previous section ;
2. find $\bar{\lambda}$, the maximum value of the truncated absolute values of the eigenvalues of $\bar{H}(V)$ and compute $s(V) = \frac{1}{\sqrt{\bar{\lambda}}}$.

Open the `vizuMet.msh` file to vizualize your isotropic metric field inside Gmsh.
Last, adapt your mesh using the metric saved in the `metric.sol` file and check the results.