MMGen2D_M_elliptic
2D elliptic and algebraic Mesh Generator - V1.1
User's Manual

Simone Marras
simone.marras@gmail.com

September 2010

Contents

1 Introduction

This is a short guide to the use of *MMGen2D*, a 2D structured mesh generator for general simply-connected domains, but thought specifically for different vertical distributions in mountain problems. It can generate quadrilaterals (QUAD) or triangles (TRI) using a Transfinit Interpolation (TFI) or an elliptic solver (elliptic). To avoid node overlapping, the elliptic method is the best option in the current version of the code, as shown in Figure **??**.

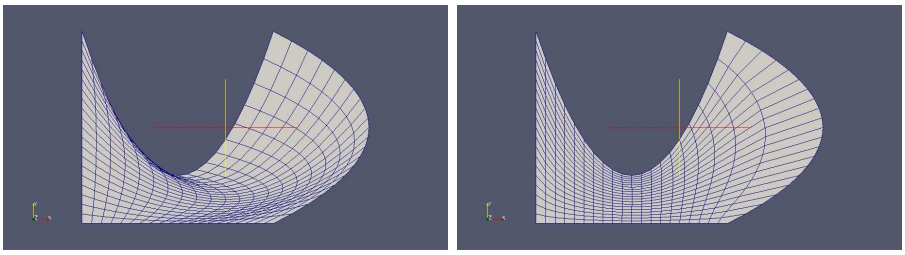

Figure 1: TFI (left) vs Elliptic.

The source code is written in C, commented throughout, and can be easily modified or enhanced by the user's own functions. I wrote this code for my own purposes but tried to write it in such a way that it could be re-usable or changed by other users. The code is free and can be modified and redistributed under the *GNU GENERAL PUBLIC LICENSE*.
The visit_writer library to write the visualization output file is included with the package. It was written by Hank Childs at the Lawrence Livermore National Laboratory for the visualization software VisIt [6], and writes grids and data into the Visualization ToolKit format *VTK*. The license of this library is reproduced in the header of the library itself and in the function *wrt2plotfile.c* that calls it.
The dynamic allocation of the arrays is done on a 1-index base through functions from the library *nrutil.c*. This function is part of the NR library and is can be freely redistributed. The NR library is a numerical library that comes with the successful volume *Numerical Recipes in C: the Art of Scientific Computing* [3].

2 Installation

The installation is standard on UNIX machines, and it was installed free of problems on a Mac OS X, 10.5.4 with gcc Version 4.0.1 and on Ubuntu Linux 8.10 with gcc Version 4.3.2. However, with this version of gcc you may encounter a warning on the definition of some strings from the library visit_writer.c. Ignore the warning because it will not

affect the correct execution of the program. Follow the steps below for installation, and go to §3 for executing.

```
1. >> cd /path/to/MMGen2D_M
2. >> cd ./src/
3. >> make clean (not necessary but suggested if you
      previously built the code on another computer).
5. >> make
5. MMGen2D_M.a should now be in the src dir.
6. >> cd /path/to/your/working/directory/
7. >> ln -s /path/to/MMGen2D_M/src/MMGen2D_M.a
```

2.1 External libraries

MMesh2D can be compiled and run without the use of any external libraries (other than *nrutil*) if topography is not read from a Network Common Data Form (NetCDF) file. Otherwise, if the input topography comes from a netcdf, the NCDF library must be installed and correctly linked. Download it from

```
www.unidata.ucar.edu/software/netcdf/
```

and install it on your system first, edit the makefile in order to link to it (see below), and add

```
#define<netcdf.h>
```

to

```
myinclude.h
```

2.2 The Makefile

The makefile is standard and straightforward to edit. If using the netcdf library simply make sure that it is properly linked according to the point of installation on your machine. By default the linking, including that to netcdf is done as:

```
$(EXE): $(OBJS)
                    $(LD) $(CFLAGS) -L/usr/include -lm -lnetcdf $(OBJS) -o $(EXE)
```

where EXE, LD, CFLAGS, and OBJS are defined at the top of *makefile*

3 Execution

It is extremely easy to run the code after compilation. All parameters are passed through 2 simple inout files: *Input_meshparam.inp* and *mountain_parameters.inp*. They are described in §4.2.

1. Have a copy or a link of the exec. MMGen2D_M.a in the working directory
2. Have your *.inp files ready inside the working directory
3. >> cd /path/to/working/directory/
4. Open the *.inp files and change the entries
5. After modification, save and close the *.inp files
6. Execute as follows:

   >> ./MMGen2D_M.a [input_file]

7. Two files should be written: *.msh and *.vtk
8. Verify the output by visualization of the *.vtk out file

4 Input Files

To run this software there is the need for one input file that must be in the working directory at the time of execution: it is called the *Input_meshparam.inp* and is described in §4.2. However, there may also be the need for a second file that must be inside the working directory only if defined in the *.inp . This file, described in §**??**, and with extension **.nc* or **.dat* or **.txt* is a file containing the height data of the real topography to be meshed. This file is only used if entry $TOPOGRAPHY \rightarrow Type$ in *Input_meshparam.inp* is set to one of the following ones: *NCDF* (for NetCDF) or *DTE* (for Digital Terrain Elevation Data).

4.1 Input *Parameters* File

The default input file that comes with this version is called *Input_meshparam.inp*. It is a text file whose entries are identified by numbers and keywords as explained in §4.2. Every change in the structure of the input file (e.g. order of the lines inside the file, addition of new entries, etc.) must be done accordingly in the function that reads this file:

`READ_INPUT.c`

Do not ever change the order of the lines inside the input file unless you modify also the function that reads it.

4.2 Structure of the Input Files

The input file has the structure represented below, where the blue string on top is the header of the file; the keywords in red identify the name of the variable whose value is assigned after the ":" ; the values (blue strings) are the only elements that can be changed by the user. A slash indicates the options that can be used. In green each entry is explained (Note: the colors are used here for the sake of illustration, but the real input a text file without colors):

```
#INPUT_FILE.inp
PROBLEM:
 name: swan/closine/square/agnesi/schar
END_PROBLEM

PROBLEM_DEFINITION:
 Space_dimensions: 3
 Meshing_scheme: TFI/elliptic/SLEVE/HYBRID/SIGMA
 Element_types: HEXA/WEDGE
 Number_blocks: 1
 nnodesx: 73 0
 nnodesy: 60 0
 nnodesz: 30 0
 PERIODICX: off
 PERIODICY: off
END_PROBLEM_DEFINITION

BDY_FILE: in_swan/in_closine/in_square/in_agnesi/in_schar

#END_INPUT_FILE.inp
```

5 Output Files

Two output files are generated, one for visualization, and one as input/initialization file for the solver developed by the user. The plotting file is written in the VTK format by the function

`wrt2VTK_unstr.c`

This function uses the library developed by Hank Childs as part of *VisIt* [6], a visualization software developed at the Lawrence Livermore National Laboratory (See the copyright notice at http://www.llnl.gov/visit/copyright.html for the redistribution of this library). The *.vtk output file can be open for visualization by any VTK reader

such as *VisIt* or *ParaView* [2].

5.1 Output *Mesh* File

The mesh file that is generated contains the list of node coordinates x_i,y_i,z_i and the connectivity matrix as depicted below. The first column (here in blue) identifies the element or coordinate numbers. The connectivity, between the keywords $ELEMENTS$ and $END\_ELEMENT$ will have 9 columns for hexahedral elements: 1 column to identify the element index (first column) and 8 columns with the connectivity nodes.

Mesh.msh file for hexahedral elements:

```
ELEMENTS
    1  1   2   399   398   157   611   108   807
    2  5   7   242     2   250   350    98   105
    3  4   8     6     3   640   125    11   197
    4  7   9     8     4   ...
    ...
END_ELEMENTS
COORDINATES
    1  0.0000 10.0000  0.0
    2  0.0000 5.0000   2.0
    3  5.0000 10.0000  0.0
    4  5.0000 5.0000   3.4
    5  0.0000 0.0000   0.0
    6  10.0000 10.0000 1.0
    7  5.0000 0.0000   1.0
    8  10.0000 5.0000  3.1
    9  10.0000 0.0000  5.2
    ...
END_COORDINATES
```

6 Mathematical fundamentals behind MMesh2D

6.1 The elliptic solver

...

6.2 TFI: transfinite interpolation

...

6.3 Vertical distribution of nodes in domains with topography

Given a proper analytical definition

$$y(x) = \{(x, y) \in \mathbb{R} \quad : \quad x_{min} \leq x \leq x_{max}\ , 0 \leq\ y\ \leq H_{domain} \quad and \quad y\ \in C^1\}$$

as the lower surface of a quadrangular and regular domain Ω in $\mathbb{R}^2$, we decompose Ω by means of non-overlapping, conformal quadrilaterals and triangles K such that

$$\Omega = \bigcup_{e=1}^{N_e} K_h\ .$$

Being h the characteristic *size* of element K, for quadrilaterals we let h correspond to the longest edge of element e, while for triangles it is defined as the length of the side that satisfies a proper property of regularity for the mesh.

6.4 The vertical coordinate system

The σ coordinate system based on height [1], the *smooth level vertical* (SLEVE) coordinates of [4], and a modified version of the hybrid pressure-based system introduced by [5] were used as vertical coordinates to test the impact that different degrees of distorsion of the elements above the topography have on the solution.

Gal-Chen and Sommerville With this transformation the topography features mapped in the computational grid decay linearly with height. The transformation $\sigma = \sigma(x, z)$ from the physical space x-z as defined above, to the space of the transformed vertical coordinate σ has the form:

$$\sigma(x, z) = H\frac{z - h(x)}{H - h(x)} \tag{1}$$

that inverted gives yields the backward mapping from the sigma vertical coordinates into the original x-y space as:

$$z(x, \sigma) = h(x) + \sigma(x, z)\frac{H - h(x)}{H}\ . \tag{2}$$

SLEVE coordinates To overcome the drawbacks of the σ transformation over steep topographies (e.g. incipient singularity of the Jacobian and its implications), the SLEVE mapping (now identified by ζ) is such that the vertical decay of the terrain characteristics reflected on the vertical coordinate depends on the scales of the topography and is not the same if different topographies are mapped. In other words, given a topographic chain, this can be obtained by the composition of a large and small scale variation (e.g. a Gaussian terrain perturbed by a wave-like function). Through this solution the vertical distorsion due to the underlying terrain is controlled from bottom to top by means of two parameters (s_i in equation 3).
The mapping is done through two successive steps: first the topography $h(x)$ is split into its large and small scale functions $h_{small}(x)$ and $h_{large}(x)$, and second h_{small} and h_{large} are combined with the decay factors

$$b_i(\zeta) = \frac{\sinh[H - \zeta/s_i]}{\sinh[H/s_i]} \tag{3}$$

to give the inverse mapping

$$z(x, \zeta) = \zeta + \sum_{i=1}^{2} h_i(x, z) b_i(\zeta) \tag{4}$$

Here ζ defines the vertical coordinate in the same way of σ in equation (1) -we called it differently to avoid confusion about the two transformations.

The combination of s_1 and s_2 drive the invertibility condition of the Jacobian (See Schar et al. for more on this).

Modified Simmons and Burridge That of [5] is basically a combination of the two methods described above. Their mapping in fact uses the same vertical coordinate σ and combines the topography $h(x)$ and the height of the domain through two functionals $a(\sigma)$ and $b(\sigma)$ whose values are properly tabled and such that the the mapping is given by:

$$z(\sigma) = Ha(\sigma) + h(x)b(\sigma)\,. \tag{5}$$

The modified version of 5 that is used next was also studied in [4], and apart from only having one only control parameter s, the structure of the inverse mapping is fully analogue to equation 4 in that it results:

$$z(x, \sigma) = \sigma + h(x)\frac{\sinh[(H - \sigma)/s]}{\sinh[H/s]} \tag{6}$$

The resulting vertical grids are exemplified in Fig.2

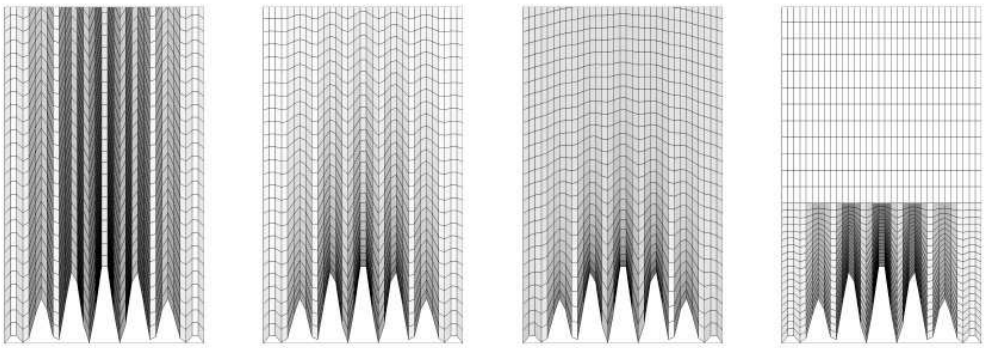

Figure 2: From left to right: σ, hybrid, SLEVE, multiblock. Shading indicates the degree of skewness of the elements.

References

[1] T. Gal-Chen and R. Somerville. On the use of a coordinate transformation for the solution of the navier-stokes equations. *J. Comp. Phys.*, 17:209–228, 1975.

[2] ParaView - Sandia National Laboratories; Kitware Inc. http://www.paraview.org/.

[3] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, 2nd edition, 1992.

[4] C. Schar, D. Leuenberger, O. Fuhrer, D. Luthic, and C. Girard. A new terrain-following vertical coordinate formulation for atmospheric prediction models. *Mon. Weather Rev.*, 130:2459–2480, 2002.

[5] A. Simmons and D. Burridge. An energy and angular-momentum conserving vertical finite-difference scheme and hybrid vertical coordinates. *Mon Wea Rev*, 109:758–766, 1981.

[6] VISIT - Lawrence Livermore National Laboratory. https://wci.llnl.gov/codes/visit/.