GESTIKULATOR
Generator of a tetrahedral mesh on a sphere

Jakub Velímský
Department of Geophysics
Faculty of Mathematics and Physics
Charles University in Prague
V Holešovičkách 2
180 00 Prague
Czech Republic
jakub.velimsky@mff.cuni.cz

April 1, 2010

Introduction

This package generates high-quality tetrahedral finite-element meshes on a unit sphere or spherical shell. It is based on recursive refinement of an initial regular icosahedron (Figure 1).

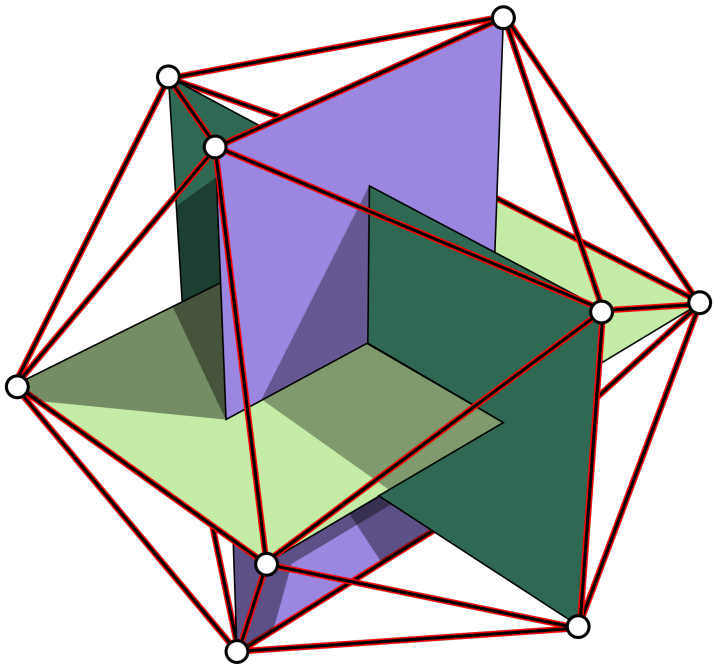

Figure 1: Regular icosahedron. (http://en.wikipedia.org/wiki/File:Icosahedron-golden-rectangles.svg, public domain).

The package consists of three programs: `gestikulator`, `locref`, and `addbottom`. Parameters for each program are set in an eponymous `ini` file which must reside in the current directory.

Description

1. `gestikulator`

   This program generates tetrahedral mesh on a unit sphere. The regular icosahedron is first split into 20 tetrahedra with 12 surface nodes and one node in the centre of the sphere. This mesh is the result of setting the level of refinement `l=0`. For `l>0` each tetrahedron of the mesh is split into eight

sub-tetrahedra (Figure 2). New nodes E, F, G, H, I, J are introduced in the centres of edges AB, AC, AD, BC, BD, CD, respectively. Each facet is triangulated into four triangles, corner tetrahedra $AEFG$, $BEHI$, $CFHJ$, $DGIJ$ are created. Finally, the internal octahedron $EFGHIJ$ is split into four sub-tetrahedra by introducing one internal edge — either EJ, FI, or GH. This procedure is repeated recursively for each tetrahedron in `l` levels. Hence, the `l`-level mesh consists of $20 \times 8^{\mathtt{l}}$ tetrahedra.

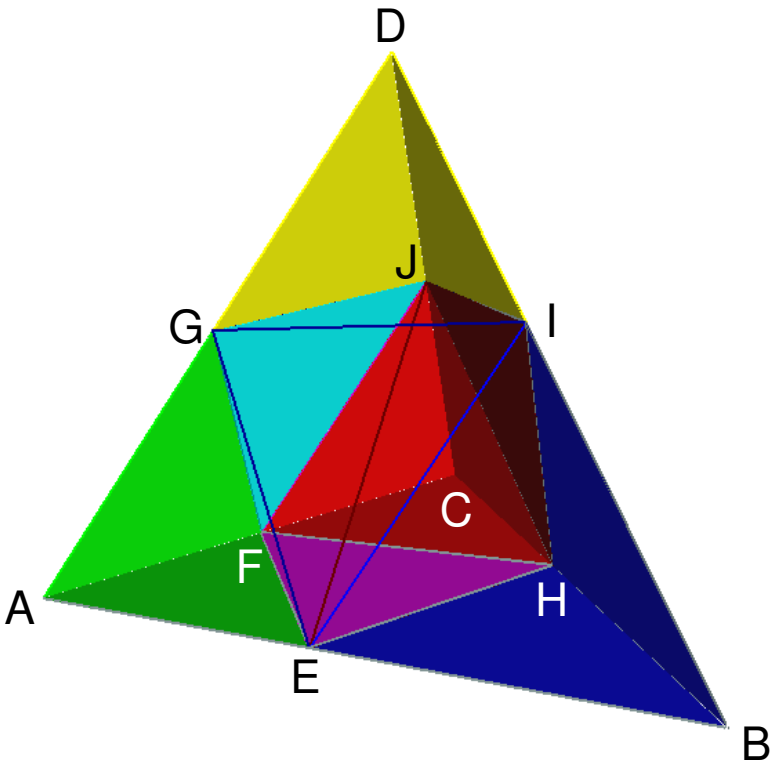

Figure 2: Splitting of tetrahedron $ABCD$ into eight sub-tetrahedra.

As the supposed use of this mesh generator lies in the field of geophysics, where many parameters are dominantly radially dependent, it is useful to create a mesh which can approximate concentric spherical boundaries. This behaviour is governed by a logical projection parameter `p`. If set to `.true.`, each node, newly created in the course of recursive refinement, is immediately projected along its radius vector to a distance from the centre, which is an arithmetic average of the radial distances of its parents. Hence, the nodes are present only at discrete spherical surfaces: `(0.0, 1.0)` for `l=0`; `(0.0, 0.5, 1.0)` for `l=1`; `(0.0, 0.25, 0.5, 0.75, 1.0)` for `l=2`; and so on. In addition, the choice of the interior edge be-

tween EJ, FI, and GH in Figure 2 is always made in such a way, that the edge does not connect nodes of different radii. Therefore, each spherical surface consists of tetrahedral facets and is completely triangulated — there are no tetrahedra that would contain volume below and above the surface simultaneously.

If `p` is set to `.false.`, this projection is not applied, except on the outermost surface — we still need to approximate the entire volume of the sphere. This leads to better mesh quality, projections along radial vectors always produce slightly deformed tetrahedra.

The syntax of `gestikulator.ini` file is self-explanatory. Description lines with arbitrary content are interleaved with the actual parameter values. Besides the refinement level `l` and projection parameter `p`, relative paths to output files have to be set. These must be enclosed in quotes, to avoid problems with spaces and other special characters. Four files describe the generated mesh in Elmer format, `mesh.header`, `mesh.nodes`, `mesh.elements`, and `mesh.boundary`. The `mesh.amf` uses AmiraMesh format (`http://www.amira.com/amira/mesh.html`). Finally, `Q.dat` contains the histogram of quality factor Q, defined as

$$Q = \frac{12(3V)^{\frac{2}{3}}}{\sum\limits_{i \neq j} L_{ij}^2},$$

where V is the volume of tetrahedron, and L_{ij} are the lengths of its edges. Note that $0 < Q \leq 1$ and $Q = 1$ for a regular tetrahedron.

2. `locref`

The second program applies local refinement to the more-less regular mesh generated by `gestikulator`. Presently, it allows to select a radius interval, where the mesh is refined. It can be used repeatedly, thickening the mesh in different domains, or even using multiple levels of refinement in the same area. For each tetrahedron, we determine how many of its vertices are inside the area of local refinement. If none, the tetrahedron is not refined. If all four vertices are in the refinement area, the tetrahedron is split into eight sub-tetrahedra using the algorithm described above. Near the boundary of the refinement zone, the tetrahedron is split into four or two sub-tetrahedra, if three or two vertices lie inside, respectively, as shown in Figure 3.

In the `locref.ini` file, one has to specify the limits of the radial interval for local refinement, the projection parameter `p`, three input files with the original mesh in Elmer format (generated by previous calls from

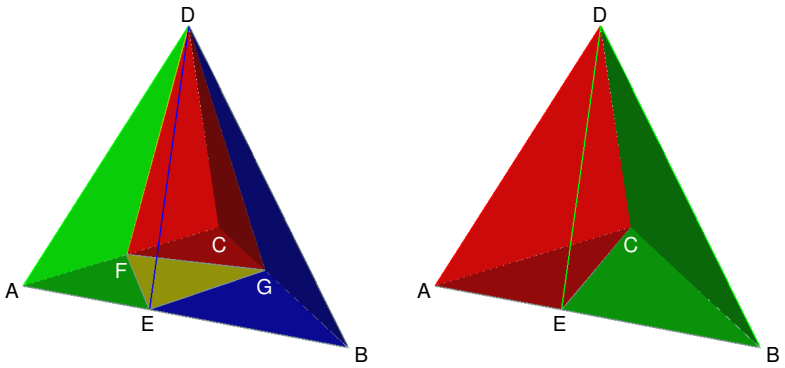

Figure 3: Splitting of tetrahedron $ABCD$ into four, and two sub-tetrahedra, respectively.

`gestikulator` or `locref`), and four output files for the refined mesh, also in Elmer format. AmiraMesh file, and Q-histogram are also generated.

3. `addbottom`

   The last program from this package takes a spherical mesh generated by `gestikulator` or `locref`, and removes an internal spherical part of the mesh, leaving only a spherical shell. The radius of the inner boundary of the shell has to be specified, the outer radius remains `1.0`. Two notes: First, the input mesh has to be prepared with projection parameter `p`. Second, the radius of the inner boundary should be close to an already existing spherical surface of the mesh. The `addbottom` program will slightly stretch it to the exact position of required inner boundary, but of course this can degrade the quality of the mesh. Use `locref` prior `addbottom` and check the resulting histogram of Q.

Compilation

1. Edit `makefile`, set up the Fortran 90 compiler.

2. Run `make`

3. Run `make doc` to create a pdf document (requires `pdflatex`)

4. Run `make examples` generate some meshes (requires `gnuplot`)

License

This package was developed by Jakub Velímský. It is distributed under the GNU General Public License v3 or later, http://www.gnu.org/licenses/gpl.html.

Modules `nrtype.f90` and `nrutil.f90` are attributed to W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, and to my best knowledge are in public domain (http://www.nr.com/public-domain.html).