BATTRI 2-D TRIANGULAR GRID GENERATOR

Version 11.11.03

Ata Bilgili*
&
Keston Smith

Numerical Methods Lab.
Dartmouth College
Hanover, NH 03755

1. INTRODUCTION

BATTRI is a graphical Matlab interface to the C language two-dimensional quality grid generator Triangle developed by Jonathan Richard Shewchuk (jrs@cs.berkeley.edu). BATTRI does the mesh editing, bathymetry incorporation and interpolation, provides the grid generation and refinement properties, prepares the input file to Triangle and visualizes and saves the created grid. Triangle is called within BATTRI to generate and refine the actual grid using the constraints set forth by BATTRI. BATTRI and Triangle are known to work on a number of platforms, including SGI's, SUN's, Pentium PC's under Linux (2.2.x and 2.4.x kernels), and Pentium PC's working under Windows. This version of BATTRI is known to work under both Matlab R11 and R12.

This report will summarize the usage of BATTRI. For Triangle usage and definitions of Triangle related files and terms, reader is referred to the original Triangle web page at http://www.cs.cmu.edu/afs/cs/project/quake/public/www/triangle.html. Help about Triangle can also be accessed by running Triangle with the -help option, i.e. "triangle -help > trianglehelp.txt".

2. DOWNLOADING AND INSTALLING BATTRI, THE OPNML TOOLBOX AND THE TRIANGLE MESH GENERATOR

To be able to use BATTRI, one should first download and install the following components:

J.R. Shewchuk's Triangle mesh generator and Delaunay triangulator can be downloaded from http://www.cs.cmu.edu/afs/cs/project/quake/public/www/triangle.html. Information on how to install and compile Triangle on various platforms can be found in the README file included in the package. For PC's working under Windows, GNU C compiler "gcc" works for compiling Triangle. You can get gcc in the latest Cygwin distribution from http://sources.redhat.com/cygwin. Triangle also includes an X display program, called

* Corresponding author: ata.bilgili@dartmouth.edu

ShowMe. Although the ability to use this program for displaying the grid existed in the previous version of BATTRI (7.2.01), it is now removed and everything is being displayed by Matlab. This means that you do not need to compile ShowMe anymore.

The most recent version of the OPNML toolbox can be downloaded from http://www.opnml.unc.edu/OPNML_Matlab. Make sure the location of the toolbox is included in your Matlab path.

Finally, the latest version of BATTRI can be downloaded from http://www-nml.dartmouth.edu/Software/BATTRI in a gnu zipped tar format. To install, gunzip (gunzip *.gz) and untar (tar -xvf *.tar) the distribution in a base directory. This will create a BATTRI directory in the base directory and copy all the files in it. The last thing that needs to be done is to add the directory location of the BATTRI to your Matlab path. This version of the BATTRI distribution does not contain any new examples. The older ones can be copied as a part of the BATTRI Version 7.2.01 distribution from the above link or from http://www-nml.dartmouth.edu/Publications/internal_reports/NML-01-2/BatTri/EXAMPLES/ .

Once Triangle, the OPNML toolbox and BATTRI are correctly installed, the user should define the location of the Triangle mesh generator explicitly. This is done by changing the triangle_path variable in generate_mesh.m routine of the BATTRI distribution.

3. INPUT FILES AND BATHYMETRIC DATA

To generate a grid, the user should input the boundary node information, boundary segment information and hole (or island) information in form of a .poly file, as described in the Triangle manual (http://www.cs.cmu.edu/~quake/triangle.poly.html). These input nodes and segments in the .poly file are forced into the triangulation of the domain. Alternatively (and this is a strong point of BATTRI), all this information can be created from only a bathymetric dataset with the use of the editing options of BATTRI (see Option 0 in Running BATTRI section). This process may require manual deleting of unnecessary segments and nodes, closing of islands by segment adding, addition of an open ocean boundary segment, etc.

As a starting point, ordered digital coastline node data can be extracted from the National Geophysical Data Center's webpage (http://rimmer.ngdc.noaa.gov/coast/getcoast.html) at various scales ranging from 1:70,000 to 1:5,000,000. If the coastline is very highly resolved, causing an excessive number of elements along the shoreline, the routine "xy_simplify.m" can be used to reduce the number of nodes to the desired resolution. Remember to format this data into a .poly file, consisting of nodes and segments, before inputting into BATTRI. To refine an already created grid, the user can input the above referenced information either in the form of a previously created .poly file or in the form of NML standard .nod, .ele and .bat files (see next section, Running BATTRI).

Bathymetric data covering the entire domain should also be entered for generation and refinement. There are four ways of accomplishing this:

As gridded bathymetric data:

>>g=generate_mesh(x,y,z)
where
z (MxN) grid of bathymetric depths, negative down from the datum.
x (1xN) x-coordinates of columns of z
y (Mx1) y-coordinates of rows of z.

As scattered bathymetric data with a pre-defined triangulation (the triangulation is used for interpolation and contouring):

>>g=generate_mesh(x,y,z,e);
where
x (Nx1) x-coordinates of depth measurements;
y (Nx1) y-coordinates of depth measurements;
z (Nx1) water depths at locations (x,y), negative down from the datum;
e (Kx3) vertex numbers for triangles in x and y.

As scattered bathymetric data with no pre-defined triangulation (Delaunay triangulation is used for interpolation):

>>g=generate_mesh(x,y,z);
where
x (Nx1) x-coordinates of depth measurements;
y (Nx1) y-coordinates of depth measurements;
z (Nx1) water depths at locations (x,y), negative down from the datum.

As a vector of multiple bathymetry data bases in which gridded (1 through j) and scattered (k) sets can coexist together:

>>g=generate_mesh(batvect);
where
batvect(1).z (M1xN1) grid of bathymetric depth.
batvect(1).x (1xN1) x coordinate of columns of z.
batvect(1).y (M1x1) y coordinate of rows of z.
batvect(2).z (M2xN2) grid of bathymetric depth.
batvect(2).x (1xN2) x coordinate of columns of z.
batvect(2).y (M2x1) y coordinate of rows of z.
…
batvect(j).z (MjxNj) grid of bathymetric depth.
batvect(j).x (1xNj) x coordinate of columns of z.
batvect(j).y (Mjx1) y coordinate of rows of z.
...
batvect(k).x (Nkx1) x coordinate of depth measurements
batvect(k).y (Nkx1) y coordinate of depth measurements.
batvect(k).z (Nkx1) list of bathymetric depth.

batvect(k).e (Kkx3) vertex numbers for triangles in x and y.

In this case, the earlier data bases take priority over the later ones. This means that if you have a high resolution local bathymetry set and a coarse resolution broad coverage global bathymetry set, list the coarse/global set last.

4. RUNNING BATTRI

The Matlab command for running the mesh generator is:

generate_mesh(x,y,z,e) or generate_mesh(batvect)

Here, x, y, z, e and batvect are as defined in the previous section with e being an optional input.

BATTRI will then displays the defined Triangle path. Make sure this directs to the correct Triangle executable. You also need to choose if you would like to use exact arithmetic with Triangle at this stage. This should always be chosen to use exact arithmetic (option 0) on anything else but the Intel architecture. On Intel platforms, we suggest trying exact arithmetic first. This usually outputs a triangulation unless you have a very complex set of points. If, however, Triangle does not produce (this is evident by Triangle hanging up during the first-cut triangulation), then you can switch to option 1 and turn off exact arithmetic. Make sure you confirm the validity of the interpolated bathymetry of the triangulation if you turn off exact arithmetic. The user is referred to the actual Triangle mesh generator web page, http://www.cs.cmu.edu/afs/cs/project/quake/public/www/triangle.html, for more information on this topic.

After the first cut triangulation of the data set, BATTRI will ask you to choose what kind of bathymetry interpolation method should be used in the entire grid generation session. The choices are linear (option 1) and grid scale dependent objective analysis (GSOA) interpolation (option 0). If linear interpolation is chosen, BATTRI continues without asking any more interpolation parameter questions. It is also important to note that linear interpolation (option 1) should be used when a vector of multiple bathymetry sets is used with BATTRI. The second option (option 0), or GSOA, assumes that the length scale in the error field covariance used in the OA is proportional to local length scales of the mesh. This way, highly refined segments of the mesh can use shorter length scales in the interpolation, while coarser areas use longer length scales, effectively smoothing the bathymetry. Two versions of GSOA are implemented in BATTRI: gsoa.m and scattered_gsoa.m for gridded and scattered bathymetry respectively. Like all OA implementations, several interpolation parameters must be specified following a series of questions that BATTRI asks if GSOA is chosen. These are:

- Standard deviation of the bathymetry measurement error (defaults to 0.25);
- GSOA covariance scaling factor, λ (defaults to 1). The relationship between the covariance (cov) and λ is as follows:

$$\mathrm{cov}_{ij}^{\ k} = e^{\frac{dist_{ij}}{L_k \cdot \lambda}}$$

Here, cov_{ij}^{k} is the covariance of length scales, $dist_{ij}$ is the distance between nodes used in the interpolation and L_k is the length scale of the grid at the node we are interpolating to.

- Number of points used in GSOA (defaults to 2 or 10 for gridded and scattered sets respectively).

It is recommended that GSOA only be used after mesh generation as it is relatively slow and prone to error due to input parameters.

You will then be prompted to enter the filename of the final grid that will be generated in the BATTRI session. This should be entered without quotes. This filename will precede the standard .nod, .ele and .bat Dartmouth NML extensions. Note that the final bathymetry file, .bat, will be positive down from the datum, unlike the input bathymetric data, which is negative down from the datum. Also note that intermediate .poly and Triangle grid files (.node and .ele) will be generated and saved every time the input .poly file is changed during a BATTRI session. These files will be saved under the filename 'finalgrid_triangle.#.*' where # is a number of iteration changed every time the grid is changed and saved during the same BATTRI session. Read http://www.cs.cmu.edu/~quake/triangle.iteration.html for more information.

You will then be prompted to enter the array of bathymetric contour values that will be drawn on the screen when the input .poly file is displayed. The format is [C_1,C_2,C_3,...,C_N] for multiple contours or [C] for a single contour, where C and C_1,...C_N are contour depths. Remember to precede these with a (-) sign if they are below the datum. This contour plotting may help user to decide what contour line should be drawn and transformed into edges whose presence will be forced into triangulation during the mesh editing session. One can also use a contour line to divide high and low resolution zones in a grid, as explained in the Mesh Editing section.

BATTRI will then ask the user to enter the minimum depth for nodes in the grid. In the final mesh with interpolated depths, any depth larger than this value will be truncated to the value of the minimum depth. For example, on a final NML grid with a Mean Sea Level (MSL) vertical datum, the smallest depth value will be -0.5 m if the minimum depth parameter is set to +0.5 m at the beginning of the mesh generation process.

The next step is to enter the name of the .poly file that you will edit or create a grid from. BATTRI gives 3 options at this stage:

- The first option (0) is to start from scratch. When given this option, BATTRI reads the previously defined bathymetric database (x,y,z) and draws the previously defined contours on the screen. One can then build the domain boundary using a given contour line in the Mesh Editing menu using option 0 without having to input a .poly file or a standard NML grid to define the boundary.

- The second option (1) is to load a .poly file that you intend to edit. You can enter the explicit path or just the name of the .poly file if it is in the current Matlab directory when prompted.

- The third option (2) is to load a previously created NML type grid by loading its .nod, .ele, .bat and .bnd files for editing. Enter the name of the grid without any extensions.

If you choose the first option (0), the program will only ask you if you would like to plot bathymetric data point locations. This is intended to be used if the bathymetric data is scattered and has highly variable density. After answering this question, it will switch to the mesh editing menu.

If you choose the second option (1), the program will ask you if there are any already defined islands (or holes) in the input .poly file. Enter 1 for yes and 0 for no at this stage. Similarly, in the next question, enter 1 if you already have defined zones (or regions) in your input .poly file or 0 if you have no zones defined. After answering these two questions, BATTRI will ask you if you would like to plot bathymetric data point locations for diagnostic purposes. At this stage, press either 1 for yes or 0 for no. As an example, this option is useful to check if your bathymetric data stays well within the limits of your boundary defined by the coastline. Assuming that the measured bathymetric data locations have the correct coordinates and datum, one can then zoom in and move coastline nodes accordingly so that the bathymetric data points will lay on the water and not on land. This is especially useful in the case of domains with a number of narrow channels. The program will then draw the boundary on the screen, together with any requested contours or bathymetric point locations and switch to the editing mode in Matlab command window.

If you choose the third option (2), the program will read the corresponding .nod, .ele and .bat files, will ask you if you would like to plot bathymetric data point locations and will switch to the mesh editing mode after you answer the question. This is also the slowest option.

5. MESH EDITING

Mesh editing menu consists of a total of 17 options, ranging from -2 to 14:

Option -2 zooms all to cover the entire limits of the domain.

Option -1 zooms in to a box defined by the left mouse button or pans using the right mouse button.

When hand editing the PSLG, interaction with the graphics window is faster if full PSLG is not plotted. Options -1 and -2 allow only the portion of the PSLG fitting inside the selected frame to be plotted. This can significantly speed up editing if the PSLG is large.

Option 0 adds a contour line to the mesh. Unlike the input contour option to generate_mesh.m, which only draws the contours on the screen, this actually adds nodes and segments whose presence will be forced in the triangulation. One can use this to mesh along contour lines (important for some hydrodynamic models with wetting and drying), to increase resolution along a contour (to better resolve a shelf or sharp changes in bathymetry) or to define various zones where different element criteria will be applied (like a higher resolution zone shallower

than the 10 meters and a lower resolution one deeper than 10 meters). Once the contour is extracted internally, the program asks user to choose between a number of smoothing algorithms. These include box-car smoothing, spline smoothing, Douglas and Peucker smoothing and no smoothing. Once the option number for the chosen algorithm and smoothing variables for different methods are entered, the program requests the user to input the node spacing (in meters) to be used when the contour will be incorporated in the grid. The user should choose a distance optimized for his/her purposes. Giving a small value highly resolves the contour and creates a large number of nodes and probably badly shaped elements too, because of resolution differences between the contour line and the rest of the grid. One solution to this is to use an element inside angle constraint (explained later), which will better the shape of these transition elements. Choosing a large node spacing results in a smaller number of nodes but may fail to resolve the contour. If a contour is going to be used to create zones (or regions) with Option 10, it is important to connect its endpoints to domain boundary by adding segments. This will ensure that the zones are going be enclosed in segments and not stay open.

Option 1 adds individual vertices to the grid using the left mouse button. Vertices can be entered one after another. Right mouse button exits from the graphical interface and returns to the MATLAB command window mesh editing menu.

Option 2 removes vertices from the grid using the left mouse button. If a vertex is connected to an edge, the edge will be moved from the domain also. Vertices can be removed one after another. Right mouse button returns to the main menu.

Option 3 moves vertices on the screen. To choose the vertex to be moved, the user should click the vertex with the left mouse button. To move the vertex, go to the new location with your mouse and left click. This process can be repeated to move multiple vertices. Right mouse button returns to the main editing menu.

Option 4 adds edges (or segments) to the grid. Segments are lines whose presence is enforced in the final grid. To add an edge, one needs 2 vertices. First choose the first vertex of the edge using the left mouse button and repeat the same thing to choose the second vertex. A red line will connect these two vertices, defining the edge. Edges can be formed one after another. Right mouse button exits to the main editing menu.

Option 5 removes edges from the grid. To do this, the user should choose the edge to be removed by clicking on the small circle whose center is located at the midpoint of the edge. The circle will be marked with a red cross. Multiple edges can be chosen one after another. To remove and return to the editing menu, hit the right mouse button. Deleting an edge does not delete the corresponding vertices.

Option 6 divides an edge into a smaller number of segments. Choose the edge to be divided in the same way as Option 5 and enter the number of pieces to divide the edge into in the command window. This process cannot be repeated and the user should choose the main mesh editing Option 6 as many times as the number of edges to be divided. This option is helpful in dividing long open ocean boundary lines.

Option 7 adds a spline curve to the grid. User has an option to choose between a closed (0) and an open (1) spline. Closed splines may be used to define simple islands (see Option 8) or zones (see Option 10) while open splines can be used to create curved open ocean boundaries. Spline nodes are entered one after another by clicking the left mouse button. Clicking the right mouse button once will draw the spline on the screen using a red line. At this stage, user can move the spline nodes around similarly to Option 3. Once one is happy with the spline shape, clicking the right mouse button will exit to the editing menu and program will ask for the desired node spacing along the spline. In choosing the node spacing, same ideas as in Option 0 apply.

Option 8 adds holes (or islands) to the grid. Even if there are enclosed areas bounded by segments in the grid, these are not treated as islands unless they they are defined as islands. An enclosed area is defined as an island by entering the x and y coordinates of a random point that lies inside the area of question. This is done by clicking the left mouse button. Islands can be added repeatedly in one session. A red cross will mark closed zones that are defined as islands. Clicking the right mouse button exits to the main editing menu. One should make sure that a zone defined as an island is actually closed by segments, otherwise, the entire triangulation will be eaten away by Triangle until an edge is encountered.

Option 9 removes holes from the grid. Holes are removed by left clicking on red crosses that define the individual islands. Multiple islands can be removed during one session. Right mouse button exits to the main mesh editing menu.

Option 10 adds zones (or regions) to the grid using the same approach as in Option 8. Zones are areas enclosed by segments where regional area constraints can be imposed on elements. Zones are defined the same way as islands, but they are marked with a green cross, followed by the zone number. Area constraints are imposed using the zone numbers at the preliminary mesh generation stage. The zone numbers start from one every time a zone adding session is started, even if there were previously defined zones. However, once zones are added and the session is closed, they are renumbered correctly automatically.

Option 11 removes zones from the grid the same way as in Option 9.

Option 12 deletes nodes and corresponding edges found in a box defined by the user. The box is defined by clicking the left mouse button on one corner of the box and dragging it until the other corner is reached. The nodes found in the box are marked with red crosses. On the command window, enter 1 if you want to remove them from the grid or 0 if you made a mistake and want to keep them in the grid. Clicking the right mouse button on the graphics window exits to the main editing menu.

Option 13 saves the current state of the domain with the editing changes to a filename.poly file. The filename is entered by the user when BATTRI prompts for it.

Option 14 exits from the mesh editing menu and saves the changes to the "finalgrid_triangle.#.poly" file.

6. PRELIMINARY (FIRST-CUT) MESH GENERATION

First-cut mesh generation generates a preliminary grid from which the refined one will be derived. Since the refinement schemes included in BATTRI are functions of element bathymetry and/or element areas, this preliminary grid serves as a base and provides the input element area and depth information that the refinement schemes will use.

The user should be careful in choosing the input parameters to the preliminary grid process and should find a good optimization between the minimum angle constraint, maximum element area constraint and the maximum number of nodes to add. Creating a very coarse first-cut grid may result in a poorly resolved domain where major bathymetric changes are missed, while creating a very fine one may result in an unnecessary excessive number of elements, increasing computation time and system requirements. Having a ballpark idea about the scale of bathymetric changes in the domain is a good place to start with in determining the input parameters. Channel-mudflat-marsh widths, characteristic lengths of major bathymetric changes like sea mounts or series of sand waves can provide some of the physical clues that the user may find useful in determining the area and inside angle constraints. The generation of a first-cut grid is an iterative process that the user can repeat in a trial and error loop if the generated preliminary grid is not satisfactory.

The input variables to preliminary mesh generation process are as follows:

Minimum angle constraint: This limits the maximum inside angle (in degrees) that an element can have in the preliminary grid. For example, if this value is set to 25 degrees, no elements will have any inside angles smaller than 25 degrees in the grid. Note that the angle constraint does not apply to small angles between input segments; such angles cannot be removed. If the minimum angle is 20.7 degrees or smaller, the triangulation algorithm is theoretically guaranteed to terminate (assuming infinite precision arithmetic, Triangle may fail to terminate if you run out of precision). In practice, the algorithm often succeeds for minimum angles up to 33.8 degrees. For highly refined meshes, however, it may be necessary to reduce the minimum angle to well below 20 to avoid problems associated with insufficient floating-point precision. The specified angle may include a decimal point. Entering a value of 0 voids this restriction.

Maximum element area constraint: This limits the maximum area that elements can have in the preliminary grid. For example, if this is set to 50,000 m^2, no elements whose area is larger than 50,000 m^2 will exist in the preliminary grid. If user has defined zones using Option 10 of the mesh-editing menu, he/she will be asked to enter different area constraints for all of the defined zones. If there are major differences between the element areas of different zones, it is likely that elements with bad aspect ratios (i.e. small angles) will be created at the common boundary of the zones. This problem can be solved by increasing or just providing a minimum angle constraint on top of the area constraints for different regions. Entering a relatively large number ensures that this constraint never gets into effect.

Maximum number of nodes to add: This is the maximum number of nodes (or Steiner points) that can be added to the preliminary grid while trying to meet the constraints of minimum angle and maximum area. Remember that this should be kept at a minimum, which is optimized

(using the angle and area constraints) to provide a nicely resolved and discretized domain with the Delaunay property. If Triangle can create a triangulation before reaching the preset maximum number of nodes, this restriction never gets into effect. If one does not want to restrict the number of nodes to add, the solution is to input a very large number and Triangle will work freely without any restrictions. Be forewarned that this number may result in a conforming triangulation that is not truly Delaunay, because Triangle may be forced to stop adding points when the mesh is in a state where a segment is non-Delaunay and needs to be split. If so, Triangle will print a warning.

Boundary Refinement: Setting this to no (option 0) runs Triangle with the -Y option. This makes sure that no new nodes are going to be added to the boundary during mesh refinement via segment splitting. Note that this will negatively affect the quality of the output grid. This option would be useful if total boundary conformity is required.

After all the input is provided, BATTRI will call the grid generation program, Triangle, and the first-cut grid will be generated and saved into the current directory. The program will then display the filenames for the new grid and some other input and output information (grid generation milliseconds, number of input nodes, segments and holes, number of output nodes, elements, edges, and boundary segments). At this stage, the newly created grid is displayed on the screen for a quick visual check by the user and BATTRI provides user with 2 choices. If you are satisfied with the grid, you should choose option 1 to continue with bathymetry interpolation onto the grid. Otherwise, you can choose option 0 and go back to regenerate the first-cut mesh.

If option 1 is chosen, the grid bathymetry (depths for newly created nodes) will be interpolated using the input bathymetry database (x, y, z Matlab column vectors). The interpolation may take a long time, depending on the number of newly created nodes. It is also normal to receive interpolation warnings at this stage if there are any nodes whose horizontal locations are outside of the bathymetry database. Depending on the results that the user derives from the diagnostic plotting routines that are explained in the next section, the preliminary grid can be regenerated if it does not meet the user's criteria by entering 0 when prompted by the program right after the diagnostics' plotting. Entering 1 proceeds to the mesh refinement step.

7. DIAGNOSTIC PLOTTING OF THE PRELIMINARY GRID

During mesh refinement it may be desirable to compute and plot characteristics of the mesh before deciding how to refine the mesh any further. To facilitate mesh exploration during refinement, the subroutine diagnostic_plots.m is called between refinements or can be called from the command line:

```
>>diagnostic_plots(g, (optional)bat);
```

where g is a finite element mesh structure and bat is the BATTRI bathymetry structure. If diagnostic_plots.m is called from the command line, the user should convert the (x,y,z) coordinates of the scattered or gridded bathymetric database to the BATTRI bat structure using

the xyz2bat.m routine (running 'help xyz2bat' and/or 'help BatTriStructs' should provide more information on this and BATTRI structures in general). Most of the plotting options implemented are for viewing which areas of the current mesh will be affected by various constraint types. The inventory of plotting options presented by diagnostic_plots.m is by no means all-inclusive. It is expected that the users will add their own options to the menu. Each plotting option launches a new figure window in which the user can zoom, rotate, and edit the plot with Matlab's figure window tools. The following plotting 11 options are supported:

h / [grad(h)*A] (Option 0): This makes a colored patch of *h/[grad(h)*A]* function on the elements of the current mesh. All variables are defined as in the mesh refinement menu (see next section). If you plan to use a *h/[grad(h)*A]* constraint in the next refinement, you can use the figure colorbar under the plot to select the value for the parameter *alpha*, so that *h/[grad(h)*A]* $\geq$ *alpha*. All elements that are colored with a color to the left of *alpha* on the colorbar will be refined by the constraint. The farther to the left on the colorbar, the more the element will be refined.

h / A (Option 1): Same as Option 0 but for *h/A*, instead *of h/[grad(h)*A]*.

1 / [grad(h)*A] (Option 2): Same as Option 0 but for *1/[grad(h)*A],* instead of *h/(grad(h)*A).*

Bathymetry Plot (Option 3): Plot the bathymetry of the mesh. This makes a colored patch object of the mesh with the coloring corresponding to the depths of the nodes. The user specifies a color axis so that they can focus on a particular bathymetric range.

Contour Comparison (Option 4): This plots the contour mesh bathymetry on top of database bathymetry. This option can be used to check that topographic features are adequately resolved by the current grid's discretization. For example, if one were trying to resolve a 10 m deep dredged channel in a harbor whose out of channel depth was no deeper than 5 m, one could contour the current mesh's 9 m isobath against that of the bathymetry databases. If the database produces two non-intersecting (blue) curves denoting the edges of the channel and the grid produces a series of elongated islands (red), then the channel has not been properly resolved by the current mesh. A gradient or slope based refinement might fix the issue by forcing more elements onto the "walls" of the channel. (You can check by using plotting option 0 or 2). This option requires the bat structure to be input if diagnostic_plots.m is manually run.

delta(h) / h (Option 5): This makes a colored patch object of *delta(h)/h* for each element. *delta(h)/h* is defined as in mesh refinement Option 6.

Grid Plotting (Option 6): This makes a simple wire-frame drawing of the current mesh.

Minimum Angle Plot (Option 7): This makes a colored patch of the element minimum inside angles. Equilateral triangles appear red, while triangles with small angles are shifted towards blue.

Element Quality Plot (Option 8): This makes a colored patch of each element's "quality" measure. The quality of an element is defined as:

$$q = 4*\sqrt{3}*A / (L_1^2+L_2^2+L_3^2),$$

where A is the area of the element and L_1, L_2, and L_3 are the length of the sides. If $q > 0.6$, triangle should be of acceptable quality. Equilateral triangles have $q = 1$. Elements with a low quality number can cause numerical problems.

CFL Required Time Step Distribution Plot (Option 9): This option uses the famous Courant-Levy-Friedrichs (or CFL) condition (i.e., $\Delta t < (\Delta x / V_{max})$, where Δt is the time step (sec), Δx is the local grid length scale (m) and V_{max} is maximum expected velocity (m/sec)) to make a colored patch of the time steps required to satisfy the condition throughout the domain. V_{max} is entered by the user when prompted by BATTRI and Δx is automatically calculated using the refined grid element areas using an equilateral triangle assumption. This option provides the user with a guideline to choose model or drogue tracking time steps that will not violate the CFL criteria.

Quit (Option 10): This quits the diagnostic plotting menu and proceeds to the mesh refinement section.

8. MESH REFINEMENT

The mesh refinement schemes of BATTRI are a collection of simple depth dependent formulae whose goal is to provide the grid generator Triangle with an array of maximum area constraints for refining individual elements. As explained in http://www.cs.cmu.edu/~quake/triangle.refine.html, Triangle is able to impose different area constraints on each element of the triangulation, besides the possibility of imposing one area constraint for all elements of the domain. This is done by creating an .area file that has one line for each element of the grid consisting of the element number and the corresponding maximum area that this specific element can have. If the area of the preliminary mesh element is larger than the one specified in the .area file, Triangle divides that element into smaller ones until the constraint is met. If it is smaller than the specified area, it is left unchanged. Elements with no constraints are marked with -999 in the .area file.

Additional control is provided through selection of a bathymetric depth range to refine. This applies a particular bathymetry based constraint only to elements which have one or more nodes with depth inside the depth range or which straddle the depth range interval. Elements outside of the depth range may still be refined due to the minimum angle constraint. If the user is interested in applying a gradient based constraint to resolve sand waves in the near shore without overly refining the shelf break, one could choose a depth range of [-40 , 0] so that the gradient constraint wouldn't be dominated by the shelf break. Choosing [-Inf, Inf] or [ ] will apply the constraint to the entire grid. Another use for the depth range parameter is to avoid singularities of the constraints. If one wants a mesh that handles wetting and drying and "0" bathymetric depth is referenced to mean low tide then inevitably the h/A constraint (or wavelength constraint) will be unbounded near the low tide line. To avoid this singularity the

h/area constraint can be applied to the depth range [-Inf, -0.5]. During the next refinement, a constant area constraint can be applied to [-.5, Inf].

When an element-by-element maximum area constraint is applied using depth dependent schemes, it is possible to have sharp transitions between elements in areas where there is a sudden and large change in domain bathymetry (i.e., continental shelf, marsh limits, etc...). Similarly to the zone refinement of the preliminary grid, this unwanted problem can be solved by increasing the minimum inside angle constraint, which will lead to a mesh with smoother transitions.

One can also impose a minimum element area constraint on grid elements. If an element area is smaller than the minimum area, BATTRI does not refine it any further, even if it violates the constraint being enforced. The minimum area restriction can be entered as an arbitrary constant or as a CFL type constant. The CFL approach uses the values of an expected maximum velocity in the domain and a time step (model or drogue tracking) entered by the user to calculate the minimum area that elements in the domain can have. Note that this approach gets into effect only in the refinement of coarser grids. If the refined elements are already smaller than the area limitation dictated by the CFL constraint set forth by the user, this will not have any effect on the output grid.

The 8 grid refinement options are:

Spring Relaxation (Option 0): The spring relaxation repositions the nodes without refining the mesh. Each node not located along the boundary is moved toward the center of mass of the polygon formed by the adjacent triangles. The effect is to make the mesh more regular, without adding any new nodes.

h/grad(h) Refinement (Option 1): This scheme uses the following formula to relate the maximum element area to the average element depth and change in depth:

$$h/[(grad(h)*alpha] \geq A,$$

where h is the absolute value of the average element depth, *grad(h)* is the absolute value of the gradient of h on the vertices of the element, alpha is a constraint ratio set by user and A is the maximum element area to be imposed.

h Refinement (Option 2): This relates the maximum element area to depth linearly via a simple factor a, according to:

$$h/alpha \geq A$$

where h is the absolute value of the average element depth, *alpha* is a constraint ratio set by user and A is the maximum element area to be imposed.

1/grad(h) Refinement (Option 3): This scheme relates the maximum element area to the change in bathymetry according to:

$$1/[grad(h)*alpha] \geq A\ .$$

Here, *grad(h)* is the absolute value of the gradient of average element depth *h* on the vertices of the element, alpha is a constraint ratio set by user and *A* is the maximum element area to be imposed.

Maximum Slope Refinement (Option 4): This scheme refines all elements whose maximum slope with respect to the x or y axes along any edge is larger than a user set value. Inputs are the threshold angle in degrees and the maximum element area required on the elements. This option may be used to resolve the effects of local small bottom disturbances like sand waves and boulders on small meshes. It is normal to have Matlab warnings of "division by zero" in this scheme. These do not interfere with the final solution since the maximum slope is filtered out and used.

CFL Refinement with Tidal Wave Celerity (Option 5): This scheme is based on the Courant-Friedrichs-Levy condition, which allows a higher grid resolution in shallower areas than deeper water zones because of the slower tidal wave celerity experienced in shallow water, given a time step. The area that an element can have is calculated using the following formula:

$$[(g*h*t^2)/R^2] = A.$$

Here, *g* is the gravitational acceleration in m/sec^2, *h* is the average depth of an element in meters, t is the model time step in seconds, *R* is the tidal wavelength to grid size ratio set by the user (default is 1) and *A* is the area that an element should have in the final grid. An element minimum inside angle constraint may be needed to top this condition to smooth the grid by creating elements with better aspect ratios.

Constant Maximum Area Refinement (Option 6): This imposes a constant maximum element area constraint on all the elements of the domain, without using the bathymetry.

delta(h)/h Refinement (Option 7): This scheme first checks to see if elements of the current grid meet the

$$delta(h)/h \leq alpha$$

criteria. Here, *delta(h)/h* is defined as:

$$delta(h)/h = (h_{max} - h_{min})/h_{min},$$

where h_{max} is the depth of the deepest node on the element and h_{min} is the depth of the shallowest. If *delta(h)/h* is less than or equal to the ratio alpha, the condition is satisfied and no refinements are performed on that element. If it is larger than alpha, the area of sub-elements in the refined mesh can be no larger than another user defined constant, 'tarea'.

Once the restrictions entered by the user are applied to the pre-cut triangulation and a refined grid is generated, BATTRI asks the user to choose from a series of diagnostic plots explained in the next section. Depending on the results shown by these plots, users can choose to end the refinement process and proceed to the generation of the final grid (Option 1), to continue refining from the current mesh (Option 0), to go back to the mesh editing and preliminary grid creation phase (Option 2) or to go back n steps to a previous version of the grid if the current one is not satisfactory at all (Option -1, -2, etc., depending on how many times the grid is refined in the current BATTRI session).

Option 2 above is especially useful if one needs the feel to change the boundary or add a zone during the refinement process. All the refinement steps that the user had gone through before going back to the editing phase are saved in memory by BATTRI and the user is asked if he wants to apply the same ones again to the new first-cut grid when the refinement phase is entered the next time. If he does, then BATTRI automatically applies all the previous refinements without the user having to reenter them again. Only a certain number of the previous refinements can be applied too, if user chooses to do so. This is done by entering the number of past refinements needed when prompted by BATTRI.

9. DIAGNOSTIC PLOTTING OF THE REFINED GRID

As in Diagnostic Plotting of the Preliminary Grid.

10. CREATION OF THE FINAL GRID

If user chooses the Option 1 at the end of the refinement process, BATTRI interpolates the grid bathymetry, reduces the bandwidth of the mesh by using either one of the Cuthill-McKee (very fast, less reduction) or Collins (very slow, more reduction) algorithms, displays the initial and reduced half-bandwidths, together with mesh properties (number of nodes, number of elements, number of boundary nodes, bandwidth) and exits. The final grid will be saved in Dartmouth NML standard mesh format under the filenames finalgrid.nod, finalgrid.ele and finalgrid.bat in the current directory ('finalgrid' is the filename of the final mesh defined in the Running BATTRI section).

11. ACKNOWLEDGEMENTS

We would like to thank B. Blanton, Alfredo Aretxabaleta, Karen Edwards, Cisco Werner (UNC), C. Denham (USGS), D. Fugate (VIMS), T. Gross (NOAA), D.R. Lynch (Dartmouth College), J. Manning (NMFS) and J. Veeramony (NRL) for their many useful contributions of code and insights. Of course, our appreciation must also be extended to J.R. Shewchuk for producing Triangle. This work was partly supported under NFS grant #97-163.

This manual is put together by Ata Bilgili (ata.bilgili@dartmouth.edu) and Keston Smith (keston.smith@dartmouth.edu). Please report all errors or comments to Ata Bilgili.