

A Simple Guide (with Examples) to Generating a  
Finite Element Mesh  
of Linear Triangular Elements Using *Battri*

by

Karen Edwards  
and Cisco Werner

Marine Sciences Department  
12-7 Venable Hall, CB# 3300  
University of North Carolina  
Chapel Hill, NC 27599-3300  
(<http://www.opnml.unc.edu>)

**Draft:** 14 July 2002

## Introduction

This manual illustrates by way of examples how to generate a finite element mesh of linear triangles using *BatTri*. It is intended to provide a quick introduction on how to use *BatTri* and is complementary to the Manual “BatTri 2D FE Grid Generator” by Ata Bilgili and Keston Smith (included in Appendix A of this document). As described in the manual by Bilgili and Smith (Appendix A):

*“BatTri is a graphical Matlab interface to the C language two-dimensional quality grid generator Triangle developed by Jonathan Richard Shewchuk. 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.”*

**The present manual is not intended to be exhaustive. It is intended to serve as a starting point for *BatTri* users using examples to illustrate the mesh-generation procedure. It should be used together with the more comprehensive manual of Appendix A where a complete description of parameter options is provided as well as the triangle website <http://www-2.cs.cmu.edu/afs/cs/project/quake/public/www/triangle.html>. Experimentation by users on the various options of *BatTri* is necessary!**

### Basic Requirements:

1. Matlab (version 5.3 or greater)
2. Installation of *FE\_toolbox* folder (available upon request - shortly it will be posted at the website <http://www.opnml.unc.edu>)

Note that *FE\_toolbox* contains utilities that are of more general use than those required by the enclosed mesh-generation procedure. For example, see Brian Blanton's description of contributed utilities at [http://www.opnml.unc.edu/OPNML\\_Matlab](http://www.opnml.unc.edu/OPNML_Matlab) and a brief introduction to available utilities at [http://www.opnml.unc.edu/OPNML\\_USERS\\_GUIDE/OUG.driver.html](http://www.opnml.unc.edu/OPNML_USERS_GUIDE/OUG.driver.html).

### Summary of steps:

1. Generate a “*poly*” file.
2. Run in a matlab environment using: *generate\_mesh*
3. Edit the mesh within matlab
4. Output the mesh data (nodes, elements and bathymetry).

What is a .poly file?

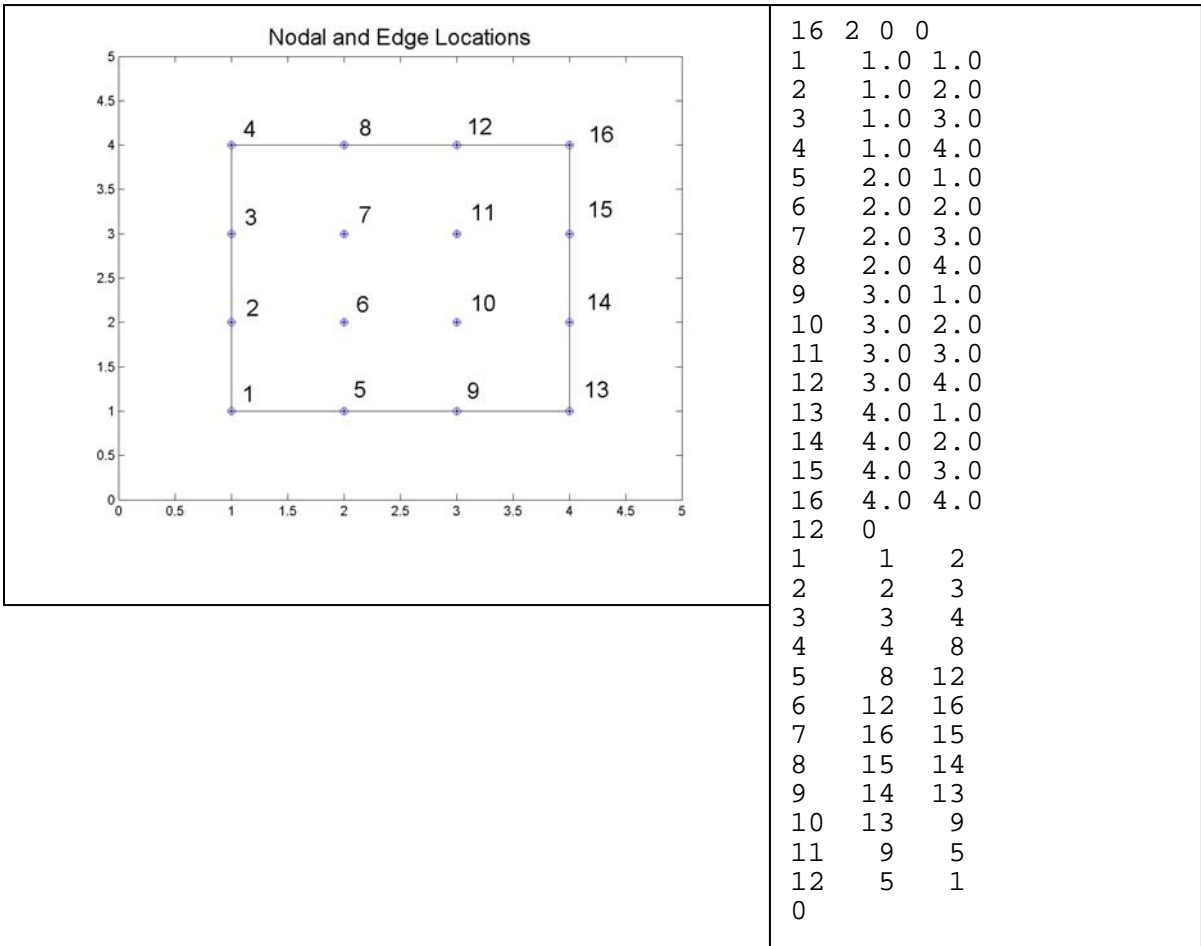
Construction of a *poly* file is the first step to the mesh generation process. It contains information of the nodal locations, boundaries of the mesh and islands (if any). The *poly* file can be generated from a digitized data base using *genpolyfile* (in preparation – not yet available), by hand, cut-and-paste, etc.

The structure of a <i>poly</i> file is:	
<b>Line 1:</b> nn nd natt nbm	nn: total number of nodes (x,y pairs) nd: # of dimensions (must be 2) natt: # of attributes (0 in these examples) nbm: # of boundary markers (0 in these examples)
<b>Next nn lines:</b> n x y	n: node number x: x-coordinate y: y-coordinate
<b>Next line:</b> nedges nbm	nedges: total # of land and open boundary element (node pairs) nbm: # of boundary markers (0 in these examples)
<b>Next nedges lines:</b> edge na nb	edge: number of the edge (or boundary line segment) na: first node of edge nb: second node of edge
<b>Next line:</b> nisl	nh: number of islands (holes) in mesh
<b>Next nh lines:</b> isl xisl yisl	isl: island (hole) number xisl: x-coord of a point that lies within the island (hole) yisl: y-coord of a point that lies within the island (hole)

See <http://www-2.cs.cmu.edu/afs/cs/project/quake/public/www/triangle.poly.html> and Appendix A for additional details.

Example poly files:

**Figure 1.** Node numbers and boundaries and associated *simple.poly* file. No island (or hole) included.





## Mesh Generation using the *BatTri* 2D FE Grid Generator – Examples

### Example 1: Simple Example using a 4x4 grid.

To start, you need to create 3 files:

<u>simple.nod</u>	contains the node numbers and the nodal coordinates
<u>simple.bat</u>	contains the bathymetric information
<u>simple.poly</u>	nodes and edges as described in previous pages

The .nod and .bat file structure conform to the NML Data File Standards as described in [http://www-nml.dartmouth.edu/Publications/internal\\_reports/NML-99-1/](http://www-nml.dartmouth.edu/Publications/internal_reports/NML-99-1/)

The simple.nod file (see Figure 1) is:

```

1  1.0  1.0
2  1.0  2.0
3  1.0  3.0
4  1.0  4.0
5  2.0  1.0
6  2.0  2.0
7  2.0  3.0
8  2.0  4.0
9  3.0  1.0
10 3.0  2.0
11 3.0  3.0
12 3.0  4.0
13 4.0  1.0
14 4.0  2.0
15 4.0  3.0
16 4.0  4.0

```

The first column is the node number, the second column is the x-coordinate and the third column is the y-coordinate.

The simple.bat file is:

```

1  0.0
2  1.0
3  2.0
4  2.5
5  0.0
6  1.5
7  3.0
8  5.0
9  0.0
10 1.0
11 3.0
12 4.0
13 0.0
14 0.5
15 2.5
16 4.0

```

The first column is the node number from simple.nod and the second column is the depth at that node.

The third file needed is the simple.poly file:

```
16 2 0 0
1 1.0 1.0
2 1.0 2.0
3 1.0 3.0
4 1.0 4.0
5 2.0 1.0
6 2.0 2.0
7 2.0 3.0
8 2.0 4.0
9 3.0 1.0
10 3.0 2.0
11 3.0 3.0
12 3.0 4.0
13 4.0 1.0
14 4.0 2.0
15 4.0 3.0
16 4.0 4.0
12 0
1 1 2
2 2 3
3 3 4
4 4 8
5 8 12
6 12 16
7 16 15
8 15 14
9 14 13
10 13 9
11 9 5
12 5 1
0
```

16 = number of nodes  
for the next 16 lines:  
node number, x-coord, y-coord.

Note that these coordinates do not have to be the same as in the simple.nod and simple.bat file but I have used those. These will be the vertexes created by BatTri

12=number of edges defined below  
for the next 12 lines:  
edge number, node 1, node 2

0=number of islands (holes) defined

Once these files are created, you can begin to run the program in Matlab. To run generate\_mesh you need to have an x, y, and z defined. In the examples that follow, **all matlab commands and responses to prompts by the programs are indicated in boldface red.**

*In Matlab (define x,y and z for the program):*

```
load simple.nod
x=simple(:,2);
y=simple(:,3);
load simple.bat
z=-simple(:,2);
```

*note that the z is defined as the negative of simple.bat column 2.*

```
generate_mesh(x,y,z)
```

*you will be asked for a series of inputs as follows:*

Enter **'simplekp'** (**BE SURE TO INCLUDE THE QUOTATION MARK IN YOUR REPOSE**) to save the final grid under this name in the current directory when prompted for where you would like the output mesh files to be saved. This may include an explicit path followed by the mesh filename if you do not want to save the final grid in the current directory. Enter the contour lines to plot while building the boundary: **[-1]**

Enter the minimum depth for nodes in the grid. This is to ensure that any depths in the database that are smaller than this will be truncated to it. For example, if your datum is mean sea level and the average tidal range is 1 meter along the coastline in the domain, enter +0.5. We entered **0**.

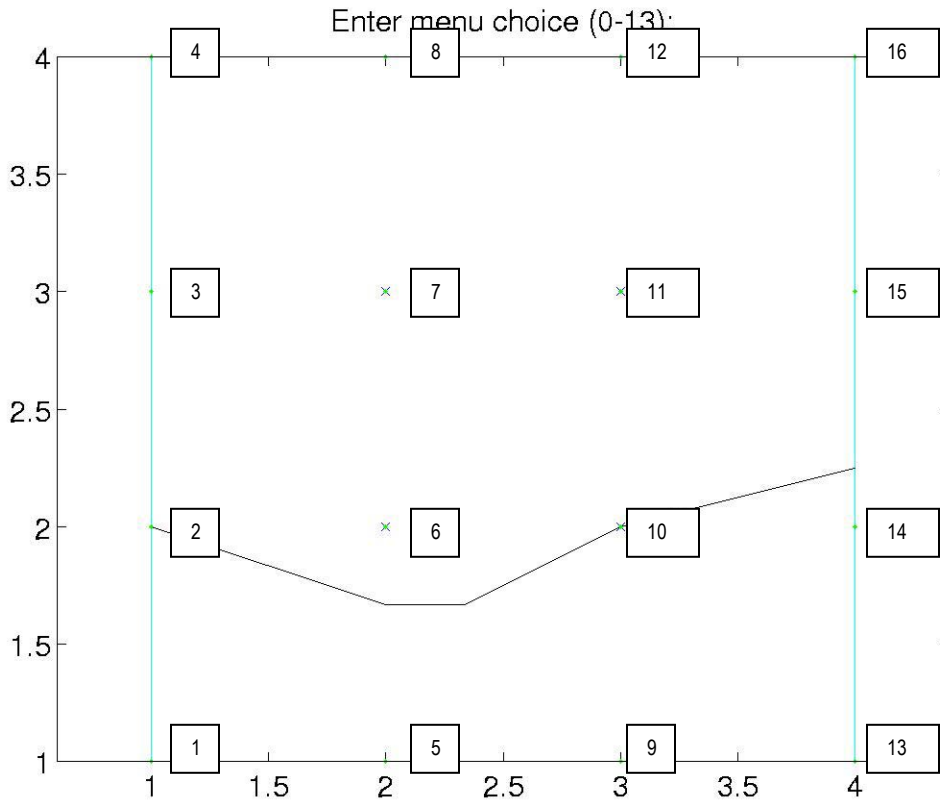
The program will then prompt you to choose between various input options. Let's choose Option **1** and enter '**simple.poly**' when prompted for the .poly filename.

When prompted, enter 1 if there are any islands defined in the .poly file (as shown in Figure 2) or 0 if there are none. If there are no islands defined, user can define them later in the mesh generating process. In our case, this is **0**.

When prompted, enter 1 if there are any previously defined zones in the .poly file (refer to Appendix A) or 0 if there are none. Zones can be defined by the user later in the mesh generation process. In our case, this is **0**.

The program will then ask you if you would like to plot the bathymetric data point locations defined in the bathymetric database given by (x,y,z). If you would like to see how the bathymetric data is located relative to the model domain, hit 1 for yes. Otherwise hit 0 for no. Let's choose **1** in this case.

At this stage, the program will display the domain on the screen (Figure 3). The grey line represents the  $-1\text{m}$  contours given as input. The green dots show the nodes containing bathymetric data. The node numbers (1-16) from the .poly file have been added. The mesh-editing menu of 13 choices will appear in the Matlab command window. You can use these to edit the input .poly file.



**Figure 3.** Nodal locations. The 1 meter contour line is indicated.



At this point, we could continue to refine and edit the domain using the menu choices 1-12. In this simple example, we are not going to do any further editing and choose **13** to get a first cut at triangulation. The newly created domain will be automatically saved to **"simplekp\_triangle.poly"** file in the current directory and program will start getting user input for the creation of the preliminary (or first-cut) mesh.

You will be asked for the following information:

Minimum angle constraint: we have used **20**

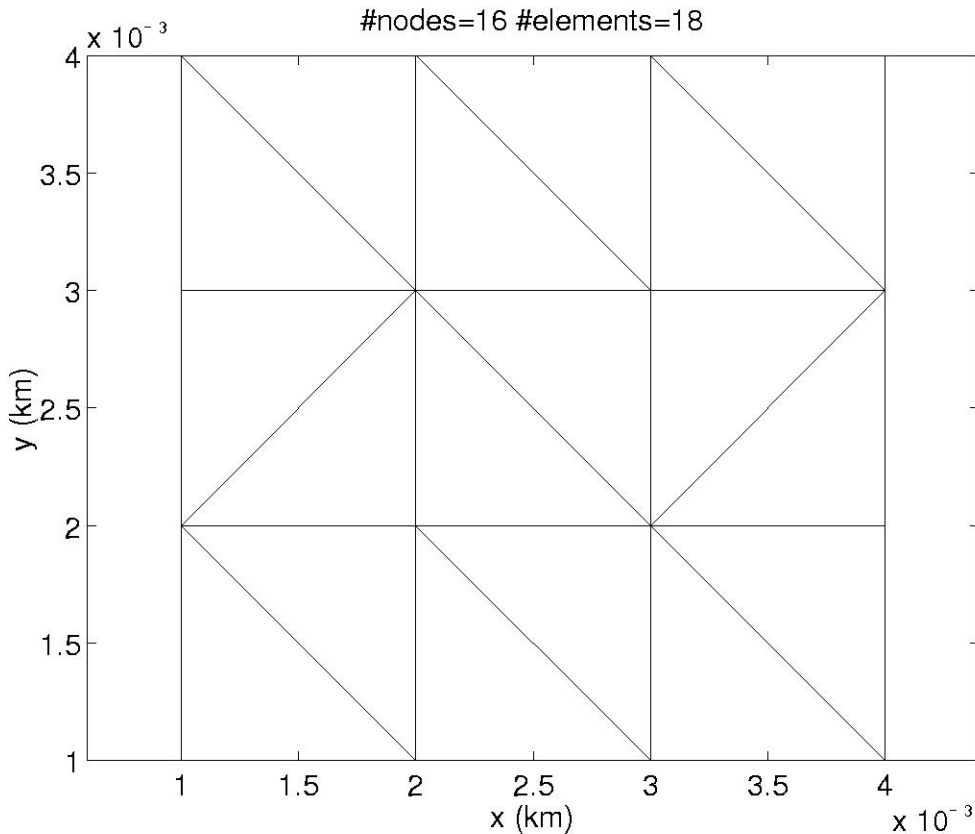
Maximum number of nodes to add: we have selected **0**

Maximum element area (m<sup>2</sup>): we have used **1** (please note that this is not a realistic value for most situations).

*If you have defined regions or zones you will be asked to input this information for each region. See definition of "region or zone" on preceding page.*

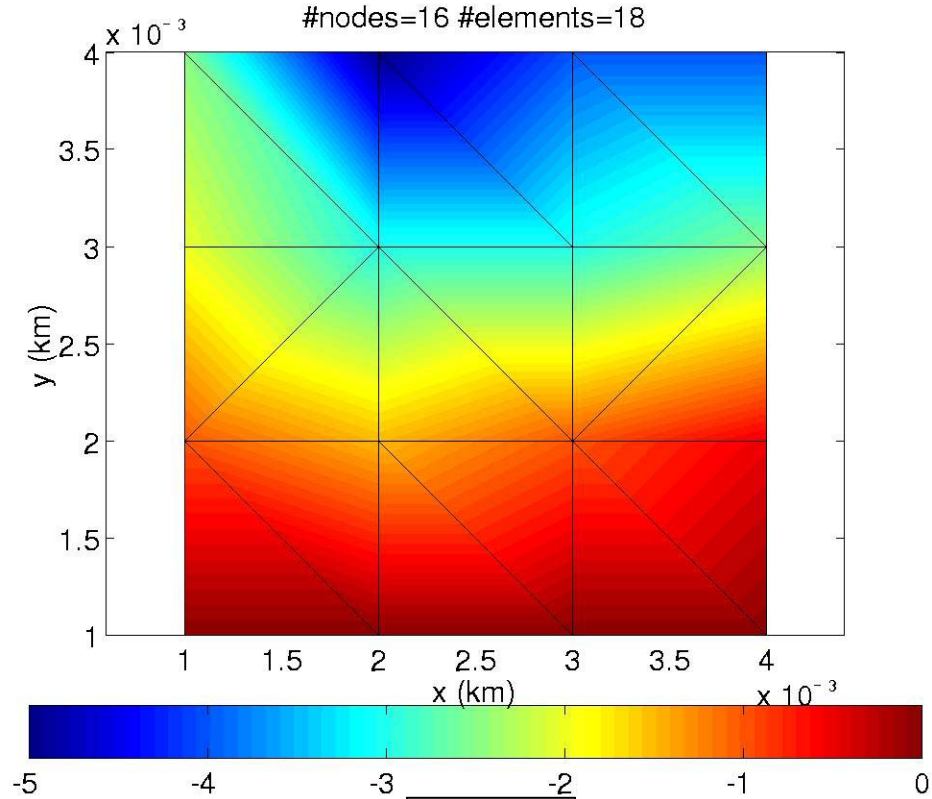
After all the input parameters are entered, *BatTri* will call *Triangle*, and automatically generate the preliminary grid, save it under filename **"simplekp\_triangle.1.\*"**, display grid properties and runtime statistics and interpolate the grid bathymetry. The interpolation may last a long time, depending on your computer properties and you may receive a number of warnings, which is normal. After the interpolation is complete, you will be greeted with a number of diagnostic plotting options.

Choose Option **6** to look at the elements in the grid. They should look like the following figure.



**Figure 4.**

Choosing Option **3** allows us to look at the grid elements superimposed on the bathymetry:



This grid is suitable as a preliminary one so let's choose **9** and proceed to the mesh refinement section. Since we are happy with the first-cut mesh, let's hit **1** and get into the refinement menu. From this menu, you have 6 different refinement options to choose from. For our simple example, this choice is not very meaningful but we have selected **3** maximum slope refinements which does not actually change the final grid from this preliminary grid. After all the input parameters are specified, *BatTri* will call *Triangle* to generate the grid as in the generation of the preliminary grid above. This time the grid will be automatically saved under filename "**simplekp\_triangle.2.\***".

After grid refinement, you are back at the plotting menu and can look at your final grid. You can then refine further or accept this grid.

*BatTri* will have created the following files:

[simplekp.bat](#)  
[simplekp.ele](#)  
[simplekp.nod](#) (These 3 files can then be loaded using the *OPNML loadgrid* command.)  
[simplekp\\_triangle.poly](#)  
[simplekp\\_triangle.1.area](#)  
[simplekp\\_triangle.1.ele](#)  
[simplekp\\_triangle.1.node](#)  
[simplekp\\_triangle.1.poly](#)  
[simplekp\\_triangle.2.ele](#)  
[simplekp\\_triangle.2.node](#)

If you load in matlab the *simplekp* grid and graph it with the node numbers included:

```
g=loadgrid('simplekp');  
drawelems(g);  
plotbnd(g);  
axis('equal');  
numnodes(g);
```

You should get something like the figure below. **The node numbers in the `.nod` file have been renumbered to minimize bandwidth but have not been changed in the `.poly` files.**

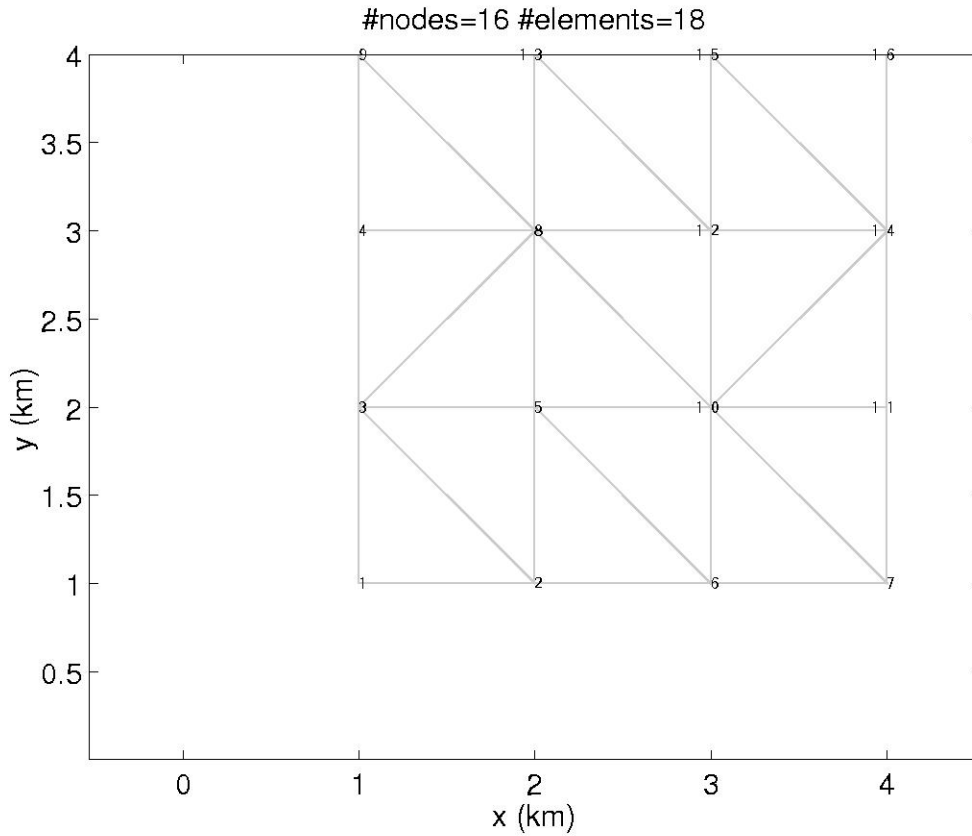


Figure 6.

**Example 2: Simple 4x4 grid with island**

Starting with the simple.nod and simple.bat files from the first example there are two ways to add an island in the middle of this grid. The first uses the same simple.poly file and then uses the editing options in `generate_mesh` to add the edges and island, which gives practice in using these editing commands. The second option is to edit the `simple.poly` file to add 4 edges and one island.

Option 1: editing in `generate_mesh`

Begin with the first example up until Fig (1). We have called our output files (*simplekp* in example 1) '*island*' in this example.

You should begin with the same grid as in Fig. (1). Using Option **4** to add edges to the grid, we want to add edges to create a square in the center of the grid (nodes 6, 7, 10, and 11). You can then use Option **8**, add holes (or islands) to the grid. Simple click in the center of the square created and a red cross will mark the island created. You should now have a grid that looks like the figure below.

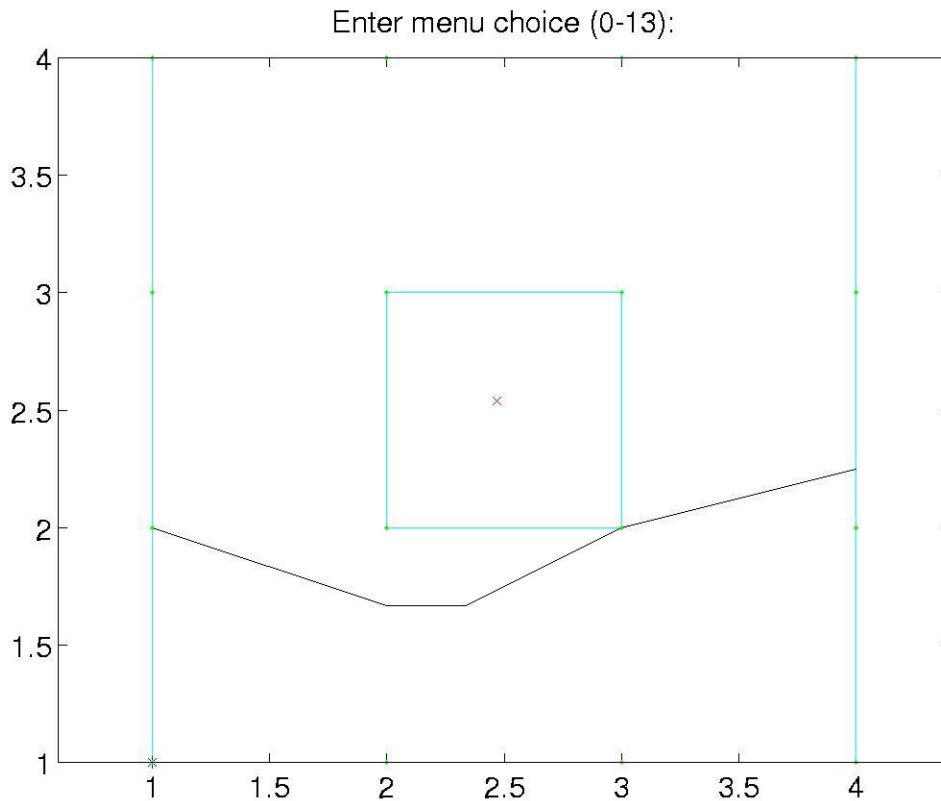


Figure 7.

We can now select **13** and generate the preliminary grid as in example 1. Choosing plot options **6** and **3** shows the following grid:

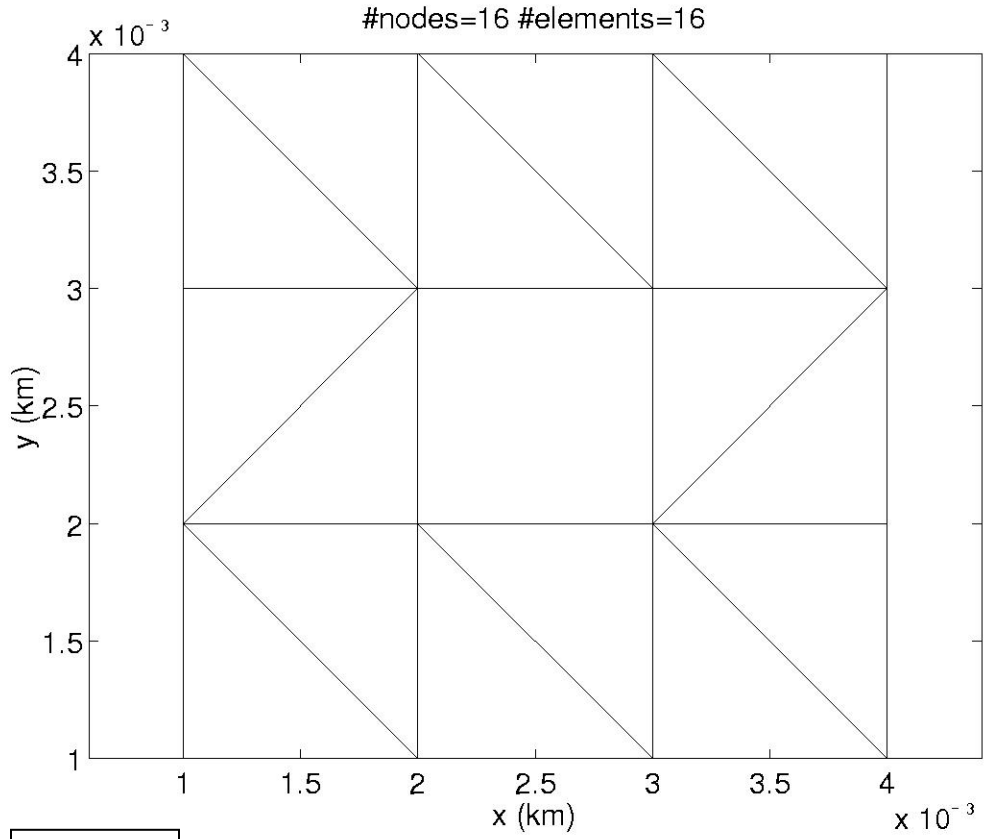


Figure 8.

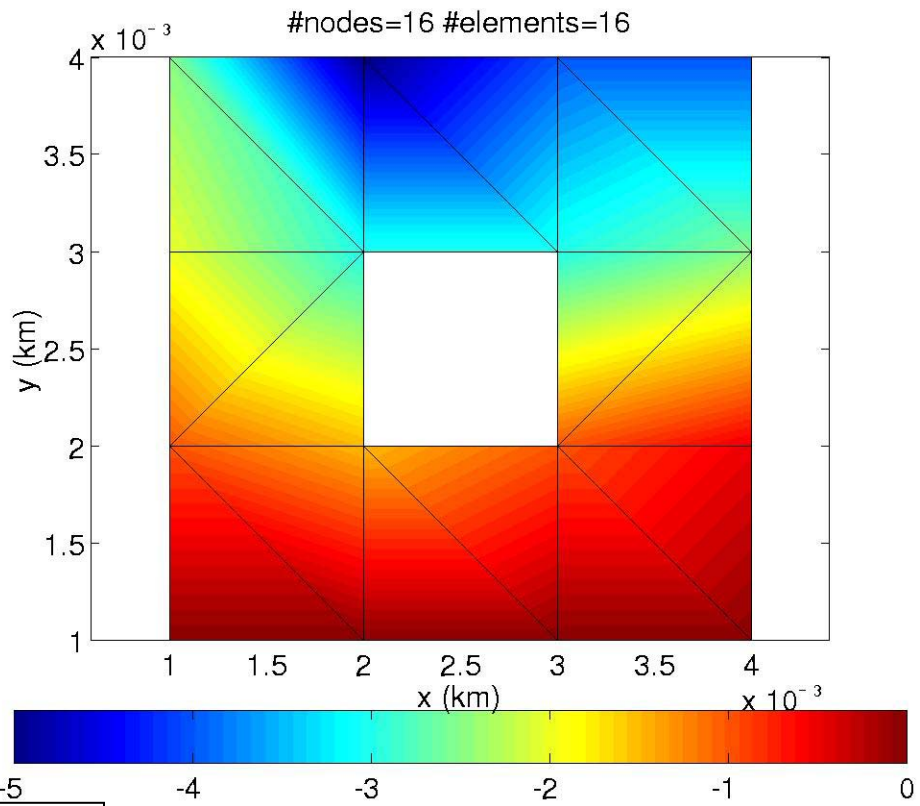


Figure 9.



**Example 3: Lake Maracaibo mesh**

Starting with an existing OPNML mesh for Lake Maracaibo, we have done two examples. The first uses the existing boundary and nodes and generates a new mesh. This turns out to be very similar to the existing mesh used in previous studies, e.g. see Figure 13 below. The second example just uses the existing boundary to generate a new mesh.

**3a: Existing Boundary and Nodes.**

Starting with the existing OPNML mesh files, included in the distribution of the FE toolbox folder:

```
mcbo.nod
mcbo.bat
mcbo.ele
```

we need to first create a boundary file and then a poly file to run *generate\_mesh*.

*In Matlab, for example:*

```
loadgrid(mcbo);
write_bnd(mcbo,mcbokp.bnd)
```

This will create a file that has:  
Column 1 = edge number  
Column 2 and 3 = nodes connecting edge.

To create the poly file (mcboall.poly), you need to use the mcbo.nod and mcbo.bnd files. The first line of the poly file should be:

```
444 2 0 0
```

There are 444 nodes in the original *mcbo* mesh, the dimension of the grid is 2, and there are 0 attributes and 0 boundary markers. You then need to copy the mcbo.nod file into the next 444 lines to provide the x,y coordinates of each node.

Following this, you need a line specifying the number of edges (161 edges, 0 boundary markers):

```
161 0
```

You then need to copy the mcbo.bnd file into the poly file to provide the edge information.

There is an island in the *mcbo* mesh, we can either define here in the poly before running *generate\_mesh* or through the editing options of *generate\_mesh*. To include it in the .poly file, the next two lines should be (again the indicated x,y coordinate is a point within the island):

```
1
1 60780.846 215956.190
```

If you choose to define the island through the editing options, you will have one line with a **0**.

Save the mcboall.poly file and we are ready to begin running *generate\_mesh*.

In Matlab (define x,y, and z for the program):

```
load mcbo.nod
x=mcbo(:,2);
y=mcbo(:,3);
load mcbo.bat
z=-mcbo(:,2);
```

note that the z is defined as the negative of simple.bat column 2.

```
generate_mesh(x,y,z);
```

you will be asked for a series of inputs as follows:

Enter **'mcboall'** (INCLUDING THE SINGLE QUOTATIONS IN THE RESPONSE) to save the final grid under this name in the current directory when prompted for where you would like the output mesh files to be saved. This may include an explicit path followed by the mesh filename if you do not want to save the final grid in the current directory. Enter the contour lines to plot while building the boundary: **[-10]**

Enter the minimum depth for nodes in the grid. This is to ensure that any depths in the database that are smaller than this will be truncated to it. For example, if your datum is mean sea level and the average tidal range is 1 meter along the coastline in the domain, enter +0.5. For this grid enter: **-1**

The program will then prompt you to choose between various input options. Let's choose Option **1** and enter **'mcboall.poly'** when prompted for the .poly filename.

When prompted, enter 1 if there are any islands defined in the .poly file (Appendix A) or 0 if there are none. If there are no islands defined, user can define them later in the mesh generating process. If you've defined the island, enter: **1**.

When prompted, enter 1 if there are any previously defined zones in the .poly file (Appendix A) or 0 if there are none. Zones can be defined by the user later in the mesh generation process. In our case, this is **0**.

The program will then ask you if you would like to plot the bathymetric data point locations defined in the bathymetric database given by (x,y,z). If you would like to see how the bathymetric data is located relative to the model domain, hit 1 for yes. Otherwise hit 0 for no. Let's choose **0** in this case as the bathymetry data is at the same points as the defined nodes.

At this stage, the program will display the domain on the screen (Figure 10). The mesh-editing menu of 13 choices will appear in the Matlab command window. You can use these to edit the input .poly file.



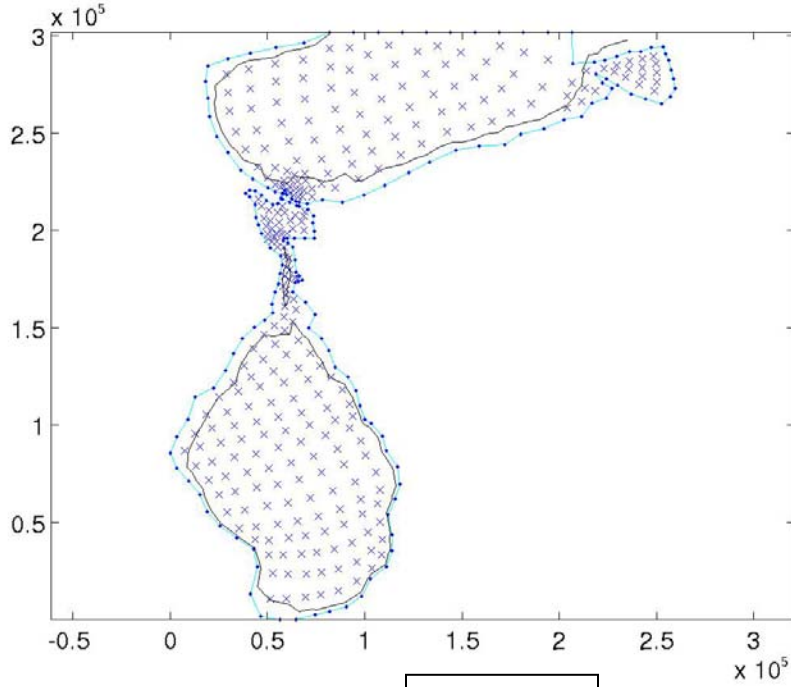


Figure 10.

In order to add the island, if not done in the .poly file, you need to zoom in on the area in Fig (11).

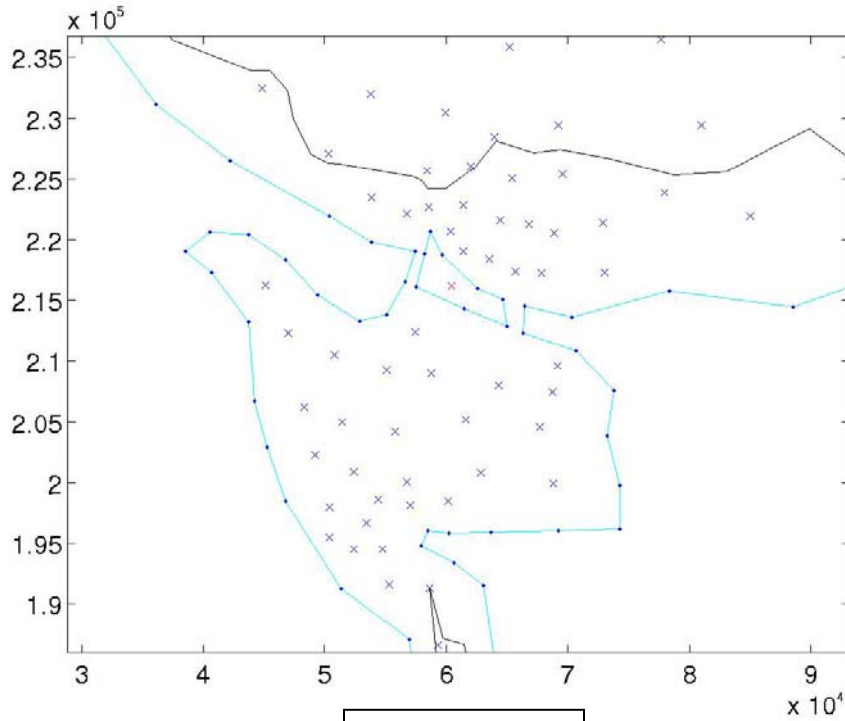


Figure 11.

Choose Option **8** add holes (or islands) from the editing menu and click in the center of the island. This will place a red x there. If you defined the island in the .poly file, there should already be a **red x** in this spot.

We are not going to do any further editing and choose **13** to get a first cut at triangulation. The newly created domain will be saved to "**mcboall\_triangle.poly**" file in the current directory and program will start getting user input for the creation of the preliminary (or first-cut) mesh.

You will be asked for the following information:

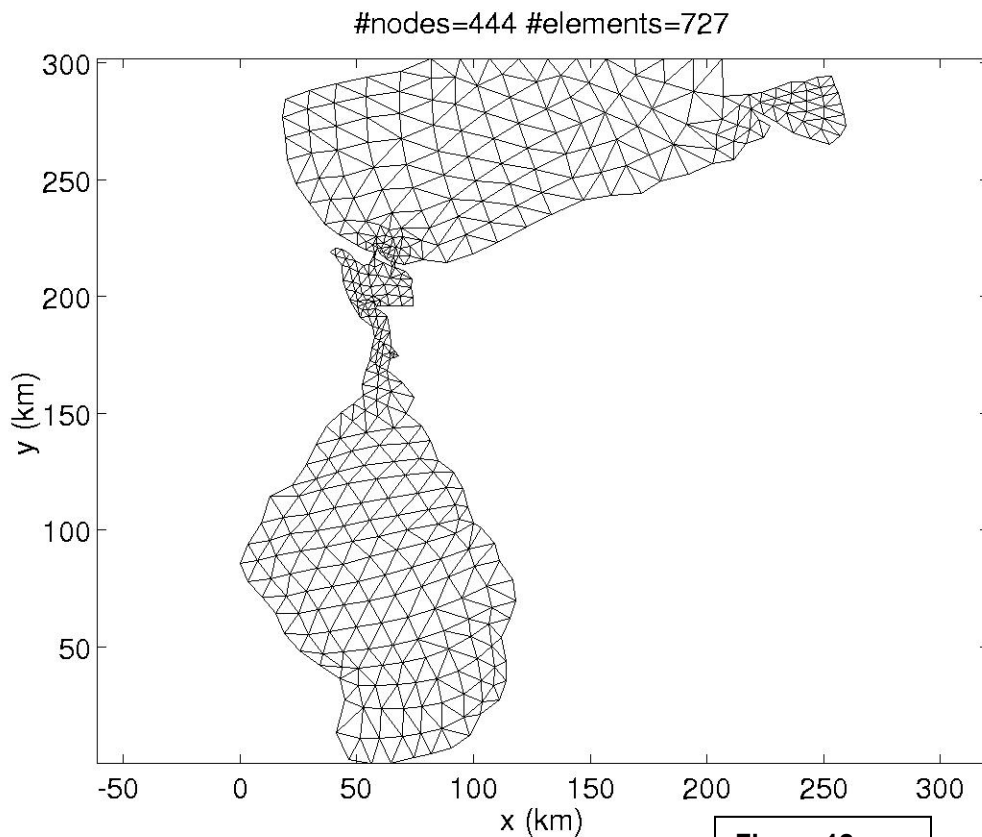
Minimum angle constraint: we have used **20**

Maximum number of nodes to add: we have selected **0**

Maximum element area (m<sup>2</sup>): we have used **250000**

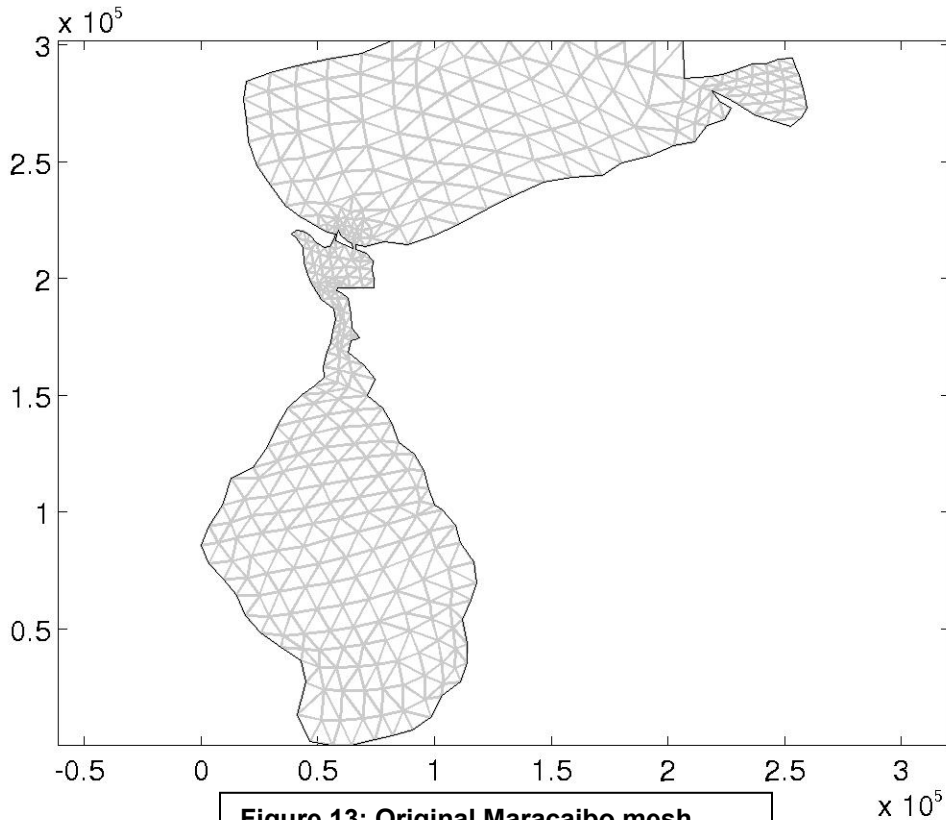
After all the input parameters are entered, *BatTri* will call *Triangle*, generate the preliminary grid, AUTOMATICALLY save it under filename '**mcboall\_triangle.1.\***', display grid properties and run-time statistics and interpolate the grid bathymetry. The interpolation may last a long time, depending on your computer properties and you may receive a number of warnings, which is normal. After the interpolation is complete, you will be greeted with a number of diagnostic plotting options.

Choose Option **6** to look at the elements in the grid. They should look like the following figure.



**Figure 12.**

This is very similar to the original mcbo mesh shown in Figure 13.



This grid is suitable as a preliminary one so let's choose **9** and proceed to the mesh refinement section. Since we are happy with the first-cut mesh, let's enter **1** and get into the refinement menu. From this menu, you have 6 different refinement options to choose from. We have selected **6** delta h/h constraint and enter:

Alpha: **5**  
 Maximum area (if violates delta h/h<5): **250000**  
 Bathymetric range: **[ ]**  
 Minimum angle constraint: **20**  
 Maximum nodes to add: **0**

My purpose in running this example was to see how close *BatTri* came to the original *mcbo* mesh, so we've again selected to add **0** nodes.

After all the input parameters are specified, *BatTri* will call *Triangle* to generate the grid as in the generation of the preliminary grid above. This time the grid will be saved AUTOMATICALLY under filename '**mcboall\_triangle.2.\***'. Since we've allowed no nodes to be added, the final mesh is the same as the preliminary mesh in Figure 12.

### 3b: Existing Boundary Only

This example is more interesting in that we will allow *BatTri* to place the nodes within the existing *mcbo* boundary. Creating the poly file for this example is more difficult in that I'm not sure if *generate\_mesh* needs the nodes numbered sequentially starting at 1. So, we've used the *mcboall.poly* and started the same as above then edited out the interior nodes.

In Matlab (define x,y, and z for the program):

```
load mcbo.nod
x=mcbo(:,2);
y=mcbo(:,3);
load mcbo.bat
z=-mcbo(:,2);
```

note that the z is defined as the negative of simple.bat column 2.

```
generate_mesh(x,y,z);
```

you will be asked for a series of inputs as follows:

Enter '**mcbokp**' to save the final grid under this name in the current directory when prompted for where you would like the output mesh files to be saved. This may include an explicit path followed by the mesh filename if you do not want to save the final grid in the current directory.

Enter the contour lines to plot while building the boundary: **[-10]**

Enter the minimum depth for nodes in the grid. This is to ensure that any depths in the database that are smaller than this will be truncated to it. For example, if your datum is mean sea level and the average tidal range is 1 meter along the coastline in the domain, enter +0.5. For this grid enter: **-1**

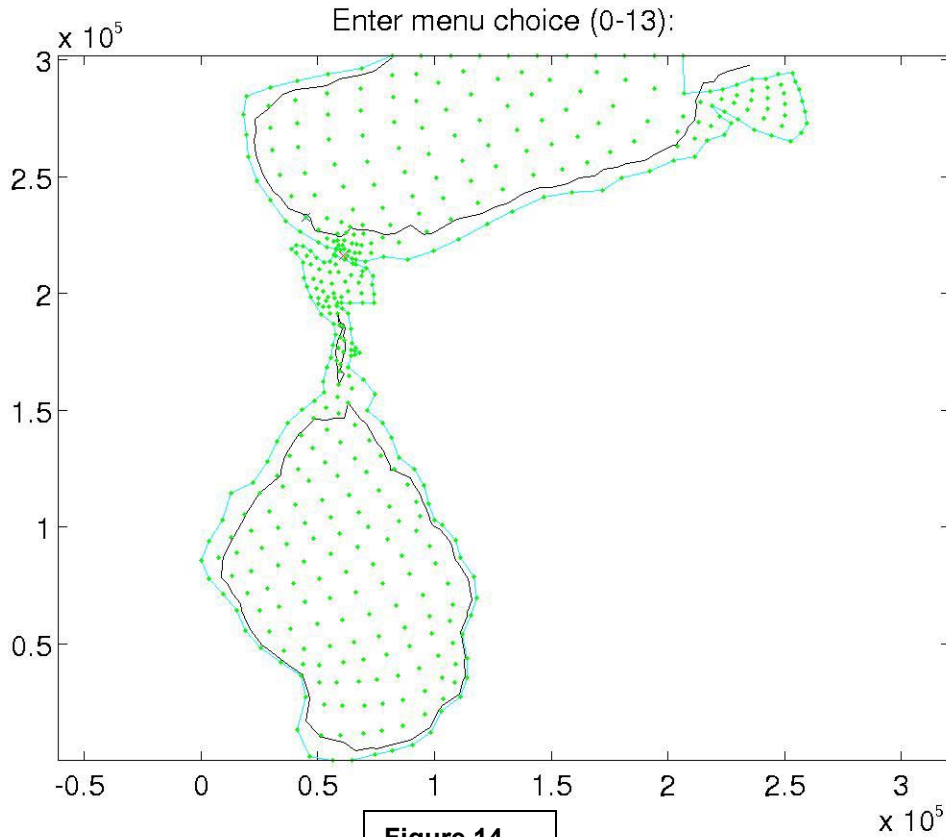
The program will then prompt you to choose between various input options. Let's choose Option **1** and enter '**mcboall.poly**' (INCLUDE SINGLE QUOTES)when prompted for the .poly filename.

When prompted, enter 1 if there are any islands defined in the .poly file (as described by *Triangle* manual) or 0 if there are none. If there are no islands defined, user can define them later in the mesh generating process. If you've defined the island, enter: **1**.

When prompted, enter 1 if there are any previously defined zones in the .poly file (as described in *Triangle* manual) or 0 if there are none. Zones can be defined by the user later in the mesh generation process. In our case, this is **0**.

The program will then ask you if you would like to plot the bathymetric data point locations defined in the bathymetric database given by (x,y,z). If you would like to see how the bathymetric data is located relative to the model domain, hit 1 for yes. Otherwise hit 0 for no. Let's choose **1** in this case as we are going to erase the currently defined nodes.

At this stage, the program will display the domain on the screen (Figure 10 above). The mesh-editing menu of 13 choices will appear in the Matlab command window. You can use these to edit the input .poly file. **By zooming in on different areas of the figure and choosing Option 12 block delete of nodes**, you can delete all of the interior nodes in the figure and get the following grid:



The green dots represent the bathymetry points (and the nodes we have deleted). But the blue line represents the boundary and the boundary nodes that we have not deleted. You can see from the red x that the island has already been defined.

Choose **13** to get a first cut at triangulation. The newly created domain will be saved AUTOMATICALLY to "**mcbokp\_triangle.poly**" file in the current directory and program will start getting user input for the creation of the preliminary (or first-cut) mesh.

You will be asked for the following information:

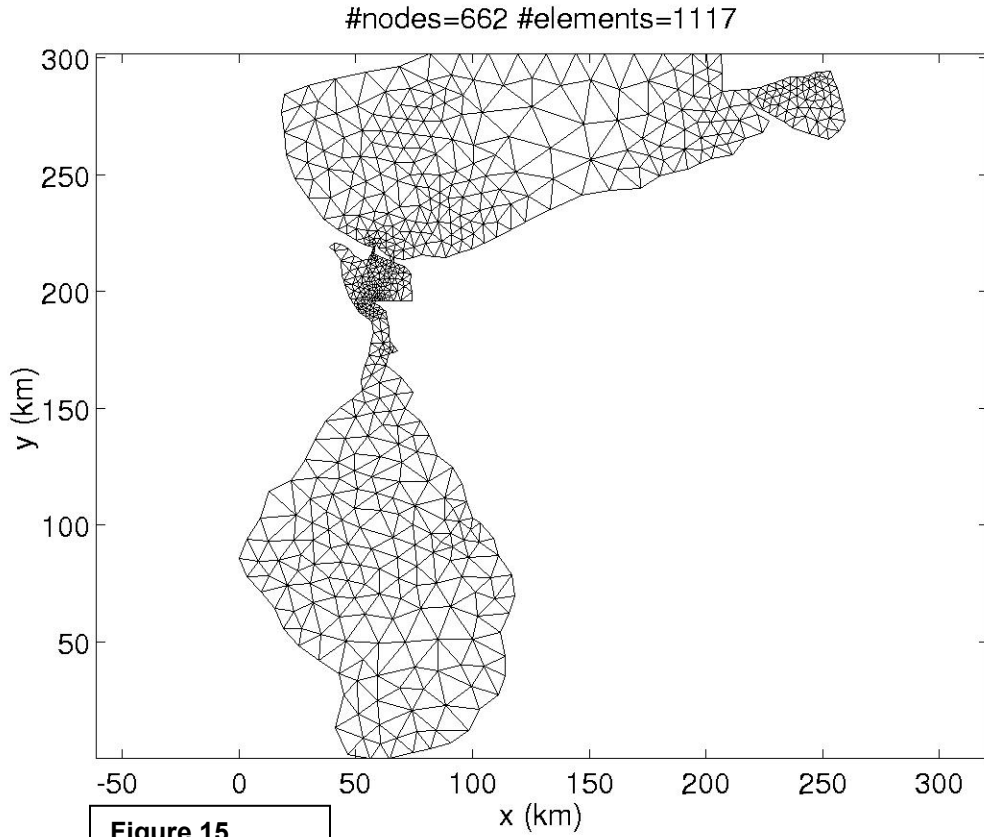
Minimum angle constraint: **20**

Maximum number of nodes to add: **500**

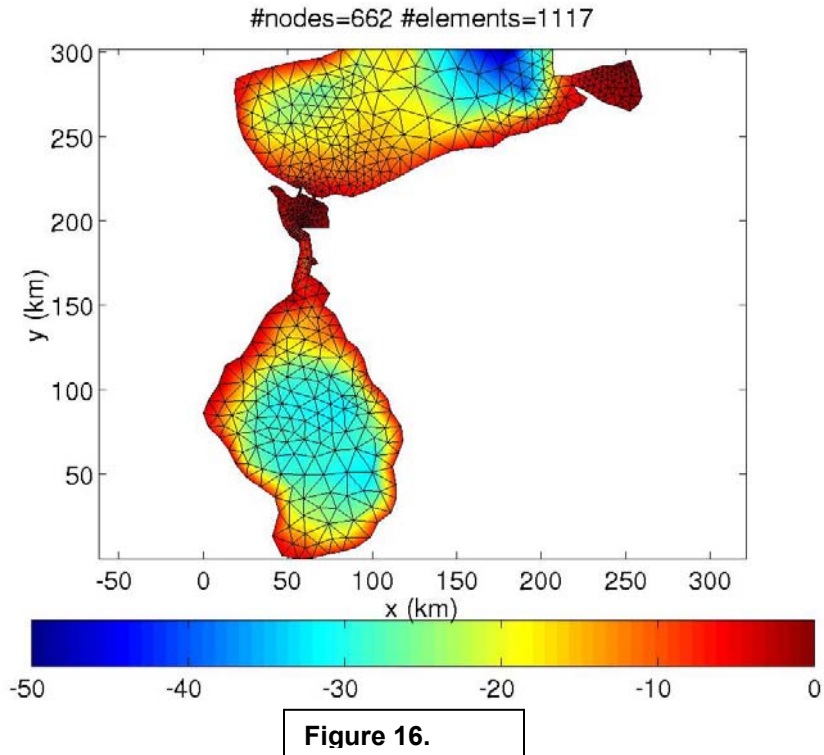
Maximum element area (m<sup>2</sup>): **250000**

After all the input parameters are entered, *BatTri* will call *Triangle*, generate the preliminary grid, AUTOMATICALLY save it under filename '**mcoall\_triangle.1.\***', display grid properties and run-time statistics and interpolate the grid bathymetry. The interpolation may last a long time, depending on your computer properties and you may receive a number of warnings, which is normal. After the interpolation is complete, you will be greeted with a number of diagnostic plotting options.

Choose Option **6** to look at the elements in the grid. They should look like the following figure.



In deciding how to refine the mesh further, we've looked at several of the other plotting options, including Options **3** and **5**:



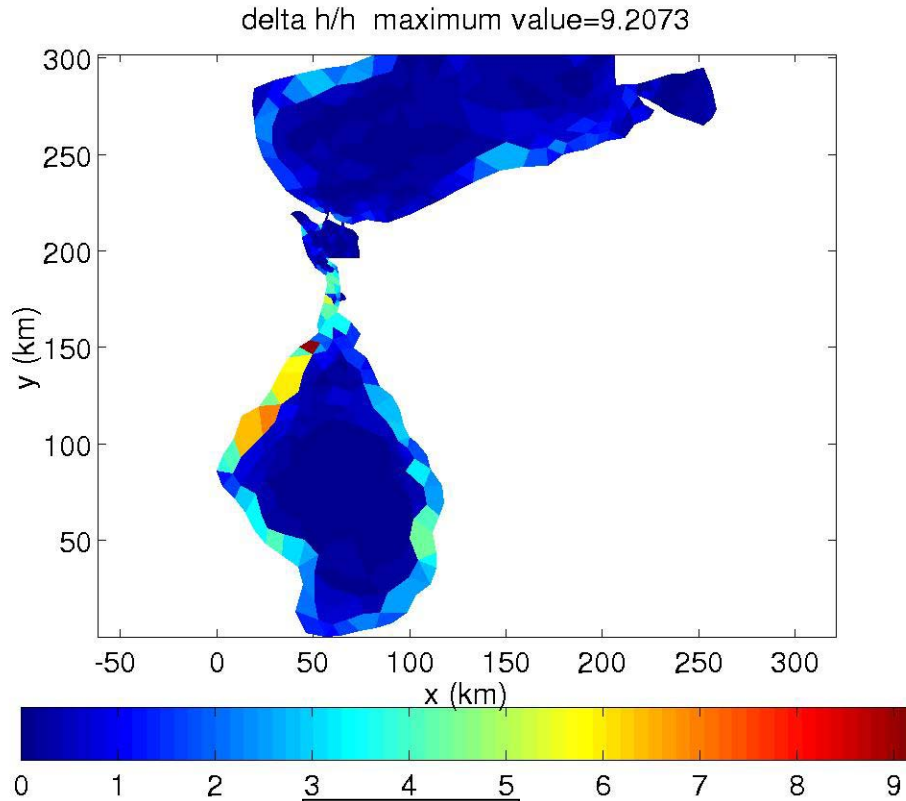


Figure 17.

We then choose **9** to finish plotting and go to the mesh refinement menu. Choosing option **6** the delta-h/h constraint which corresponds to the plotting Option 5 shown in Figure 17 above, we provide the following input:

Enter alpha: **5**  
 Maximum area (if delta-h/h<5 violated): **100000**  
 Bathymetric range: **[ ]**  
 Minimum angle constraint: **20**  
 Maximum nodes to add: **100**

After grid refinement, back at the plotting menu, we choose Option **6** again to see the final grid:

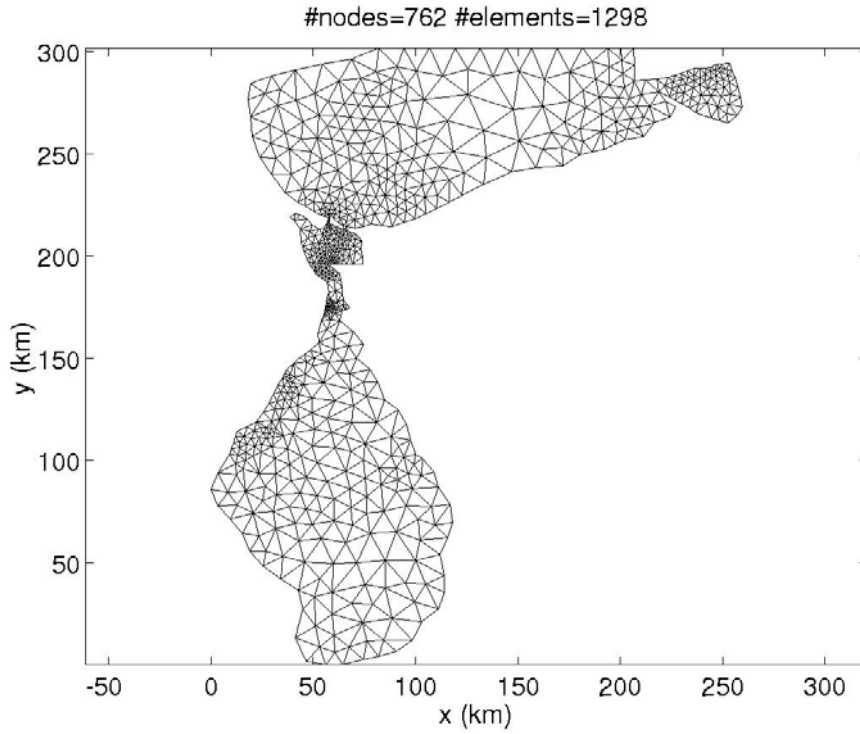


Figure 18.

During refinement, the program has added the maximum number of nodes, 100, so it is possible that not all of the other criteria have been met and that we should rerun this and add more nodes.

Since, Option 5 shows us the criteria that we have used to refine the mesh, here is the new picture of  $\Delta h/h$ :

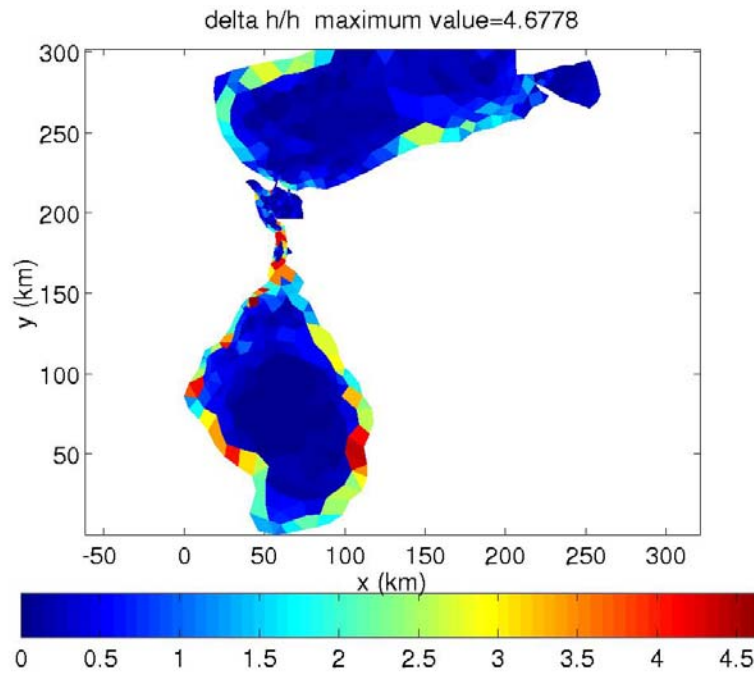


Figure 19.



As you can see, the maximum value of delta  $h/h$  has dropped to 4.6778 from 9.2073 in the initial grid (Figure 17).

Whether or not this is the best criteria to be used for refining this mesh, is to be determined. We picked it from looking at the different plots available. We also did not consider whether the other constraints such as minimum angle and maximum area are optimal. These are merely provided as an example of what *BatTri* can produce.

## Creation of a .bel file: running *genbel4*

To run the Fundy/Quoddy model suites (<http://www-nml.dartmouth.edu/circmods/gom.html>), a .bel file is needed to include boundary condition information in the mesh. A description of the contents of a .bel file is available at [http://www-nml.dartmouth.edu/Publications/internal\\_reports/NML-99-1/NML-99-1.html#bel](http://www-nml.dartmouth.edu/Publications/internal_reports/NML-99-1/NML-99-1.html#bel).

**Note: *genbel4* is not part of the BatTri code. It is part of the OPNML finite element Matlab utilities generated by Brian Blanton (<http://www.opnml.unc.edu/>). It is described here as it provides a graphical way of generating the input files needed to run the Fundy/Quoddy suite of models after the mesh generation has been completed.**

### Description of .bel file (from the above NML website)

.bel: A boundary element description of the boundary conditions. There is a two-line header:

line 1: the geometric meshname

line 2: arbitrary text identifying the contents fo the file

Following these lines, there are NBE lines of the form:

$$\text{LIN}(1,L)\text{IN}(2,L)\text{IBC}(\text{left})\text{IBC}(\text{right})$$

where:

element L is a line segment beginning at node IN(1,L) and ending at node IN(2,L)

IBC(1) indicates the nature of the boundary on the left of the element

IBC(2) indicates the nature of the boundary on the right of the element

IN(I,L) is a conventional incidence list for 1-D, linear boundary elements

L indexes from 1 to NBE, where NBE is the number of boundary elements

The node numbers refer to the nodes in the triangular mesh file meshname.nod, identified in the first header line. The boundary codes are as given in the FUNDY4 Users' Manual (Lynch, 1990):

0: interior

1: land

2: island

3: nonzero normal velocity

4: geostrophic outflow

5: elevation

6: corner: elevation with land or island

7: corner: geostrophic with land or island

8: corner: nonzero normal velocity with land or island

9: corner: elevation with geostrophic

10: corner: both components of velocity = zero

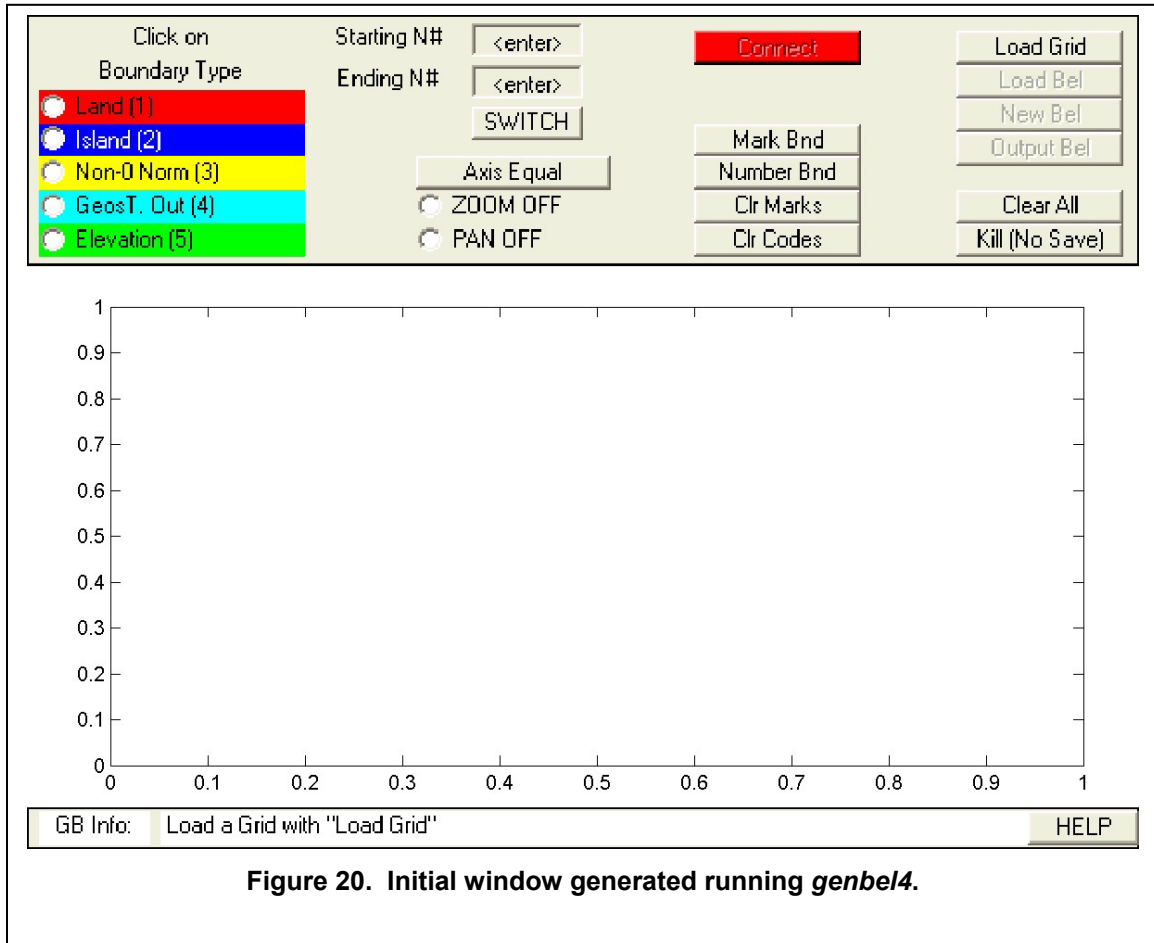
11: corner: both components of velocity nonzero

Only codes 0-5 are required to describe boundary elements; the additional corner codes are needed only for boundary node classification; they never appear in .bel files.

In a matlab window, type:

**genbel4**

The following window (Fig. 20) should appear:



**Figure 20. Initial window generated running *genbel4*.**

In this example we will impose boundary conditions on the *mcbo* mesh. Select "**Load Grid**" and specify the input *.nod* and *.ele* files. In this case, *mcbo.nod* and *mcbo.ele*.

Since there is no existing *.bel* file at this point, click on "**New Bel**", and then on "**Axis Equal**". You should see the following display (Fig. 21):

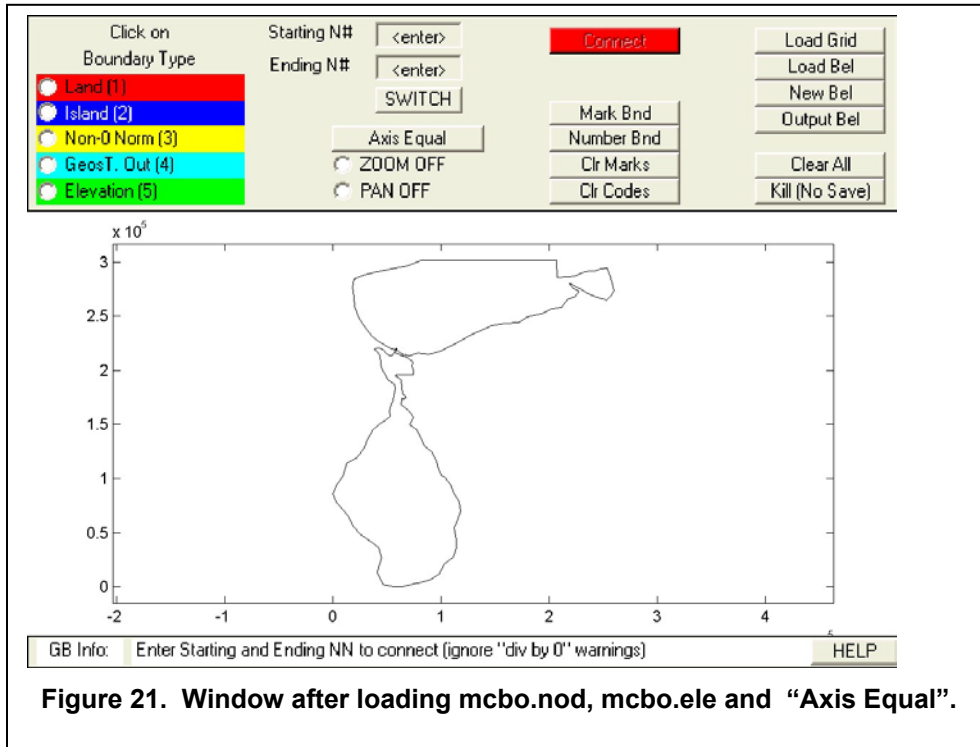


Figure 21. Window after loading mcbo.nod, mcbo.ele and “Axis Equal”.

The northern boundary of the mesh is an open boundary and thus an “Elevation” boundary condition (type 5) needs to be specified. To see the node numbers along the open boundary, under “Tools” in the upper toolbar, click on “Zoom”, choose the area of the zoom graphically. Follow this by clicking on “Number bnd” and a figure something like (Fig. 22) should result:

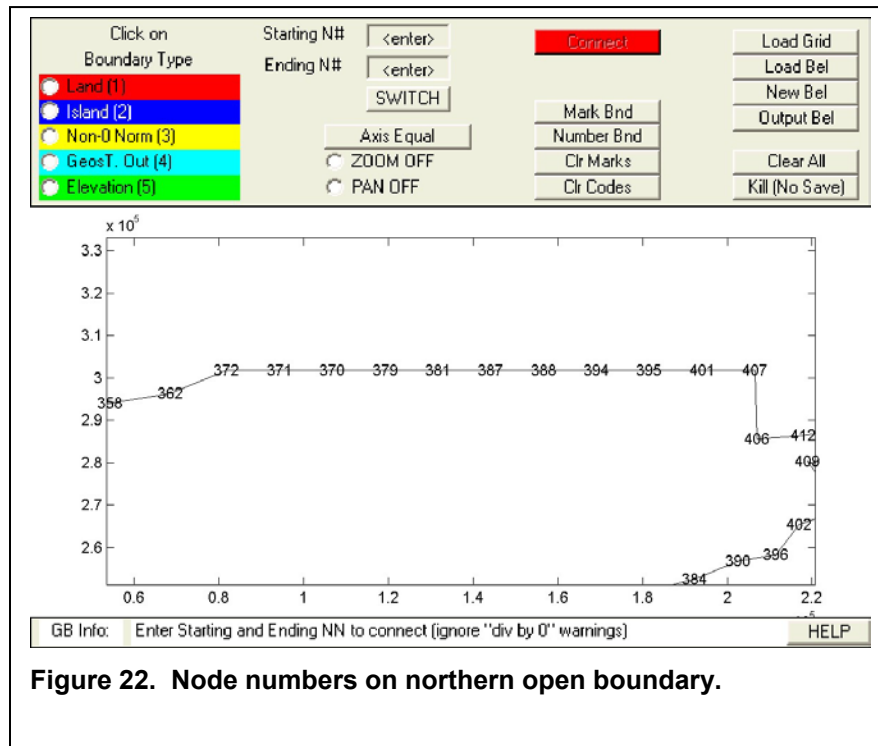
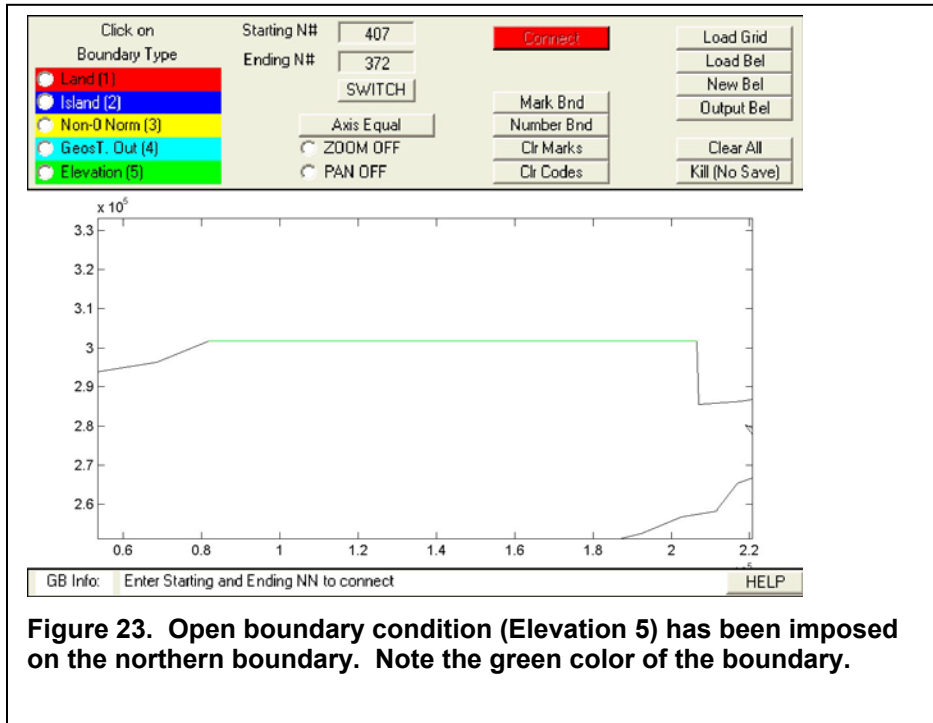
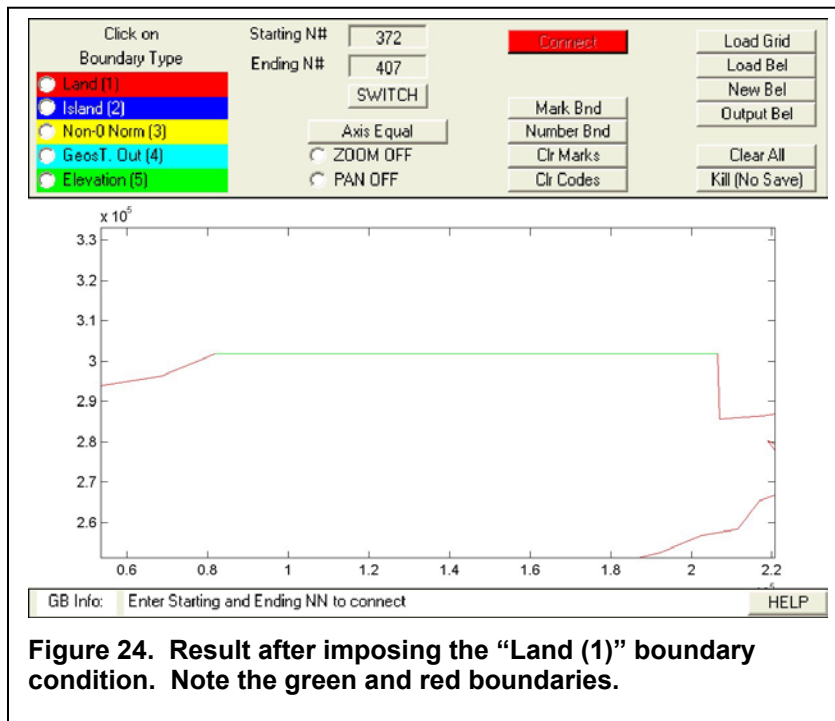


Figure 22. Node numbers on northern open boundary.

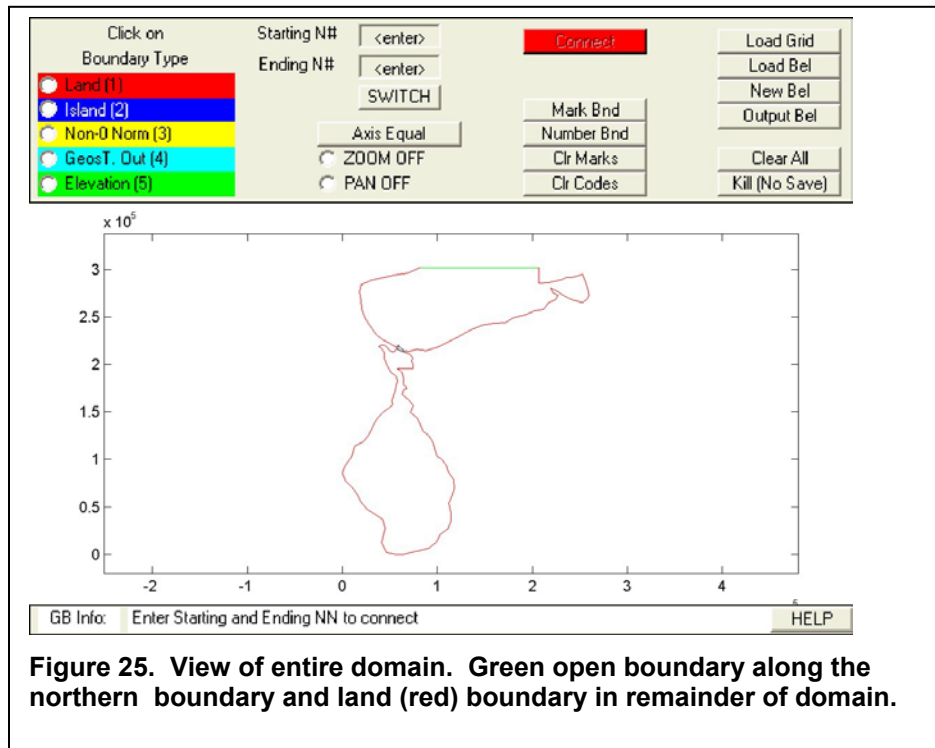
Specify the starting and ending node 407 and 372 respectively in the available boxes, then click on “Elevation 5” and finally on “Connect”. The following (Fig. 23) should result:



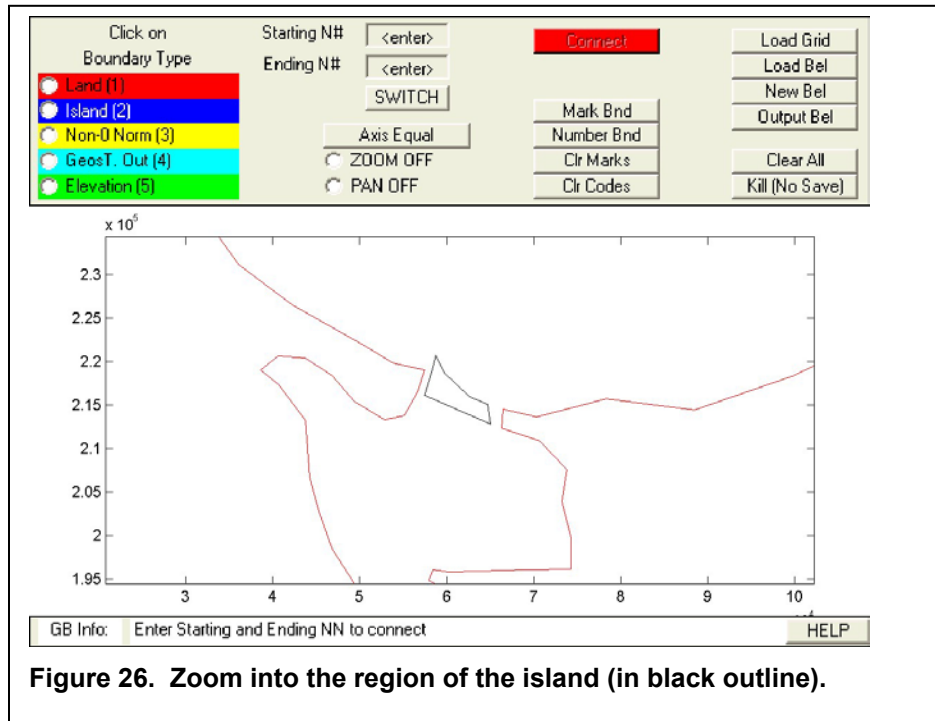
To specify the land boundary, click on “Land (1)”, click on “Switch” and finally on “Connect”. The result in Figure 24 should result:



To view the entire domain, zoom out and the following (Fig. 25) should be obtained:



To add the island, zoom into the region of the island and obtain (Fig 26):



Zooming in even more and clicking on “**Number Bnd**” results in Fig. 27:

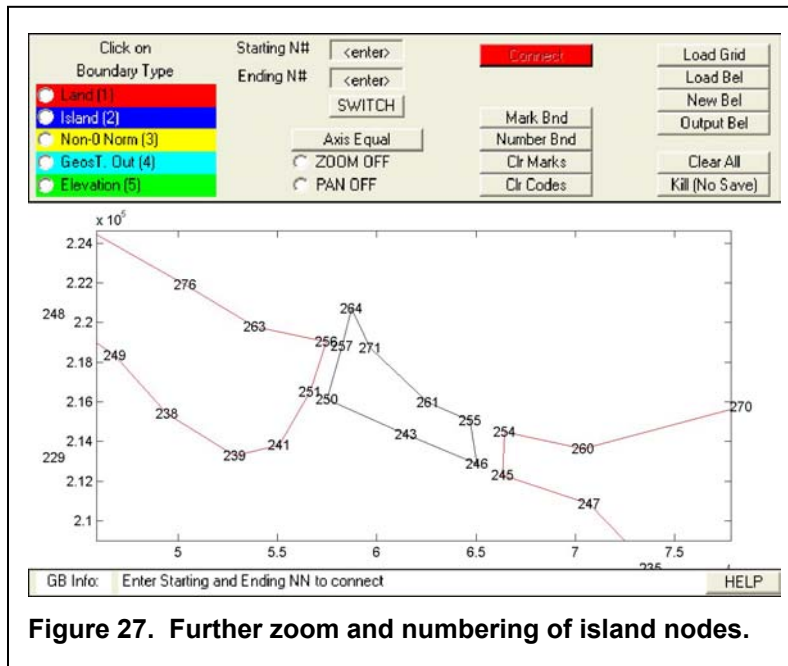


Figure 27. Further zoom and numbering of island nodes.

To specify the island boundaries, enter 264 and 271 in the “**Starting N#**” and “**Ending N#**”, click on “**Island (2)**” and then click on “**Connect**”. To complete the island, click on “**SWITCH**” and click on “**Connect**” again. You should see the following (Fig. 28) result:

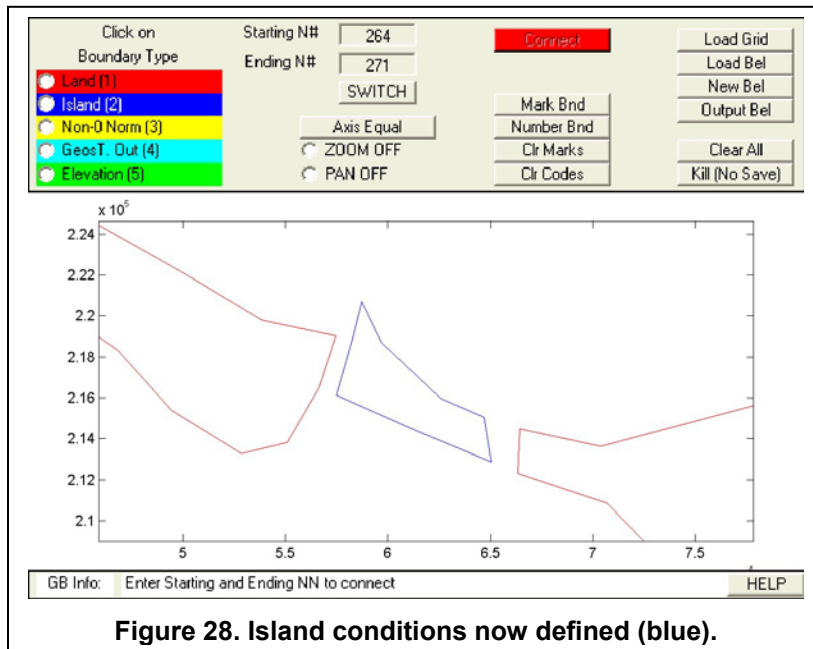


Figure 28. Island conditions now defined (blue).

To finish the creation of the `.bel` file, click on “**Output Bel**” and answer the prompts in the window that will pop out. Under “Domain name” enter `mcbo`, under “Comment” enter any comment (this will be ignored by the circulation models), under “.bel filename” enter `mcbo.bel`. Other boundary conditions such as “Geostrophic (4)” would be generated in a similar fashion.

## Chopping an existing mesh

This set of utilities is also not part of the *BatTri* code. These utilities were developed by Brian Blanton and are not yet formally part of the OPNML finite element Matlab release package. In the following example, we will illustrate the chopping of the Gulf of Venezuela (the northern basin of the *mcbo* mesh from Lake Maracaibo proper (the southern basin)).

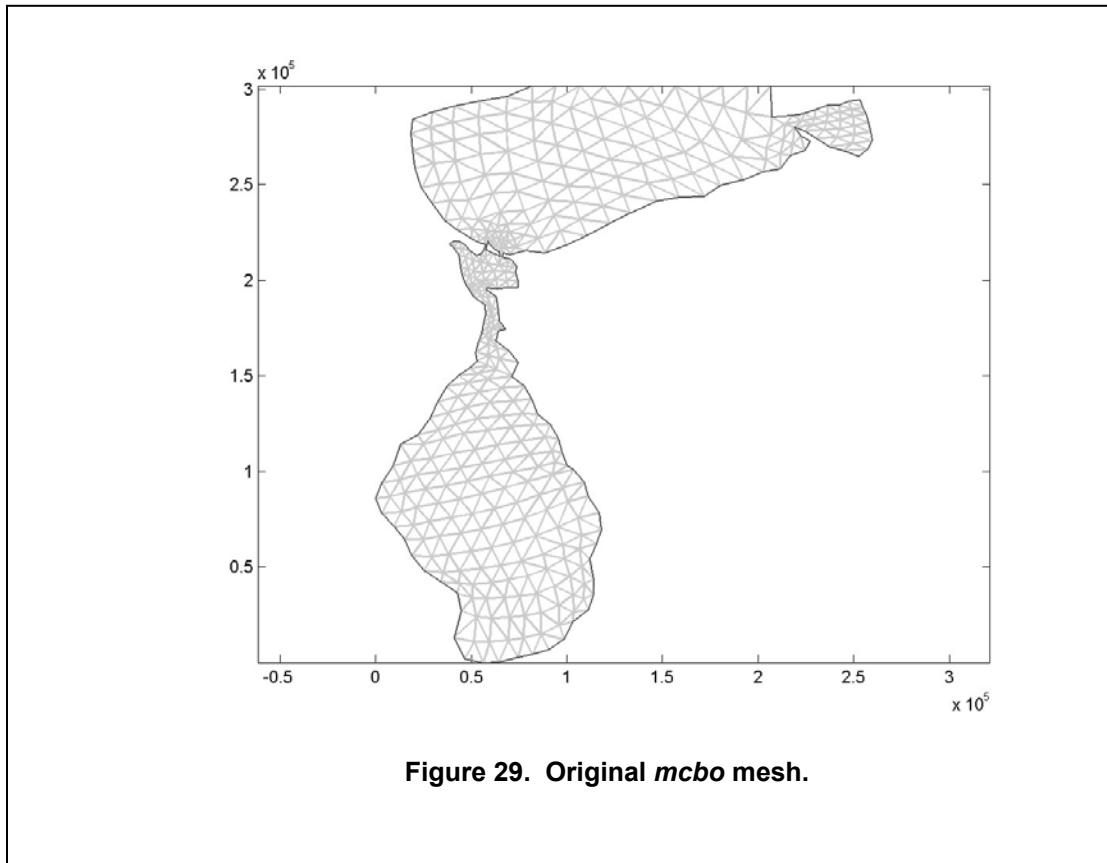
1. load a `fem_grid_struct` and attach element finding information

```
mar=loadgrid('mcbo');  
mar=belint(mar);
```

2. draw original mesh

```
drawelems(mar)  
plotbnd(mar)  
axeq
```

The result is shown in Figure 29.





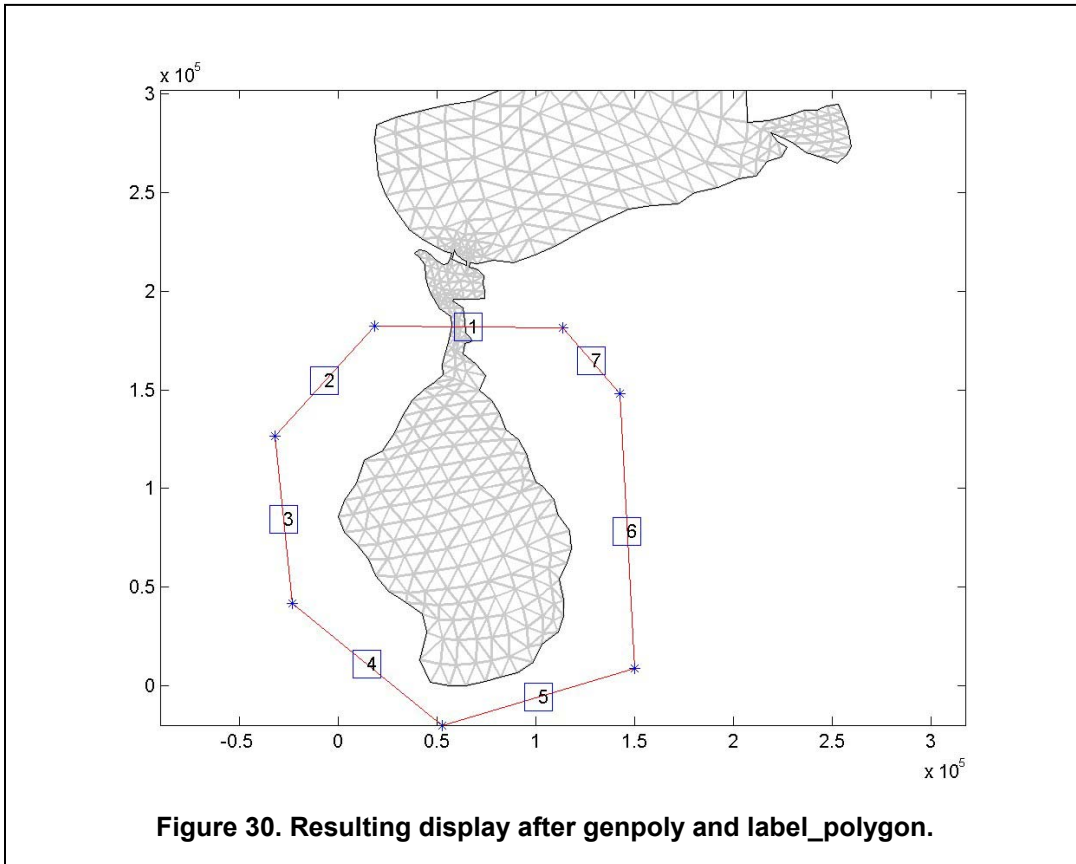
3. Generate a polygon that is closed, and defines the region of the domain that is to remain. *genpoly* is mouse-driven

```
p=genpoly;  
label_polygon(p)
```

The following response will appear in the window:

*Click on vertices with left mouse-button, right button to end*

The following figure shows a possible selection (where the blue \* correspond to locations where the mouse-clicks took place and the numbers correspond to the boundary segments).

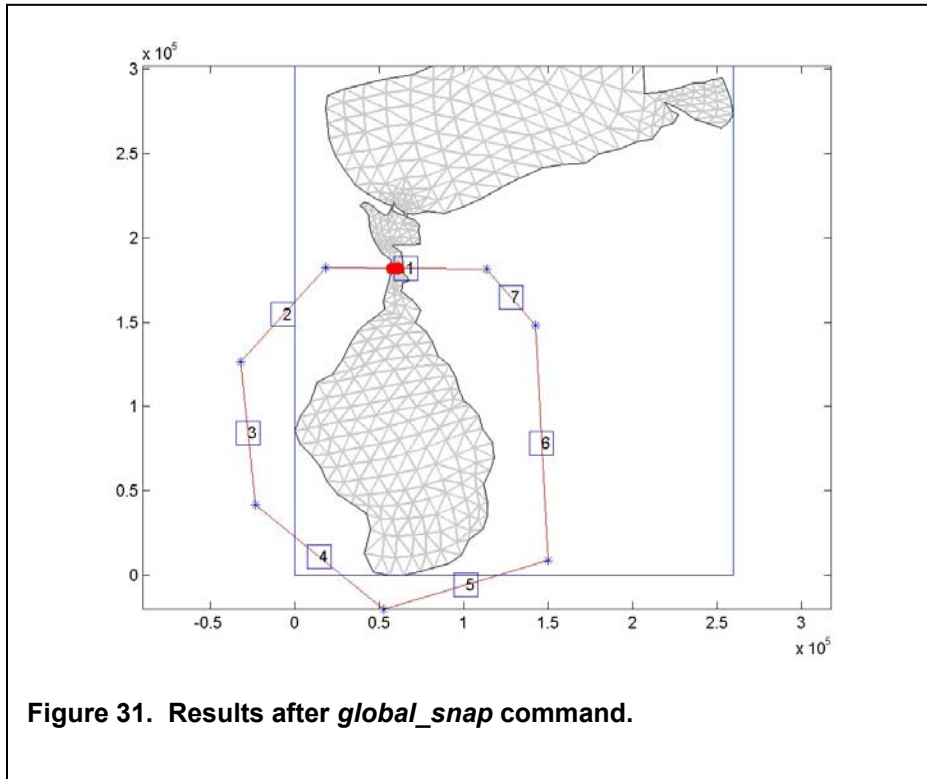


4. Move nodes in the interior of the domain onto the polygon segments. Depending on grid size, this can take quite a while.

```
mar2=global_snap(mar,p);
```

The following response and Figure (31) are obtained:

```
Processing polygon segment 1  
Pass 1...  
Pass 2...
```



5. Extracts the interior/exterior of the polygon as the new fem\_grid\_structs

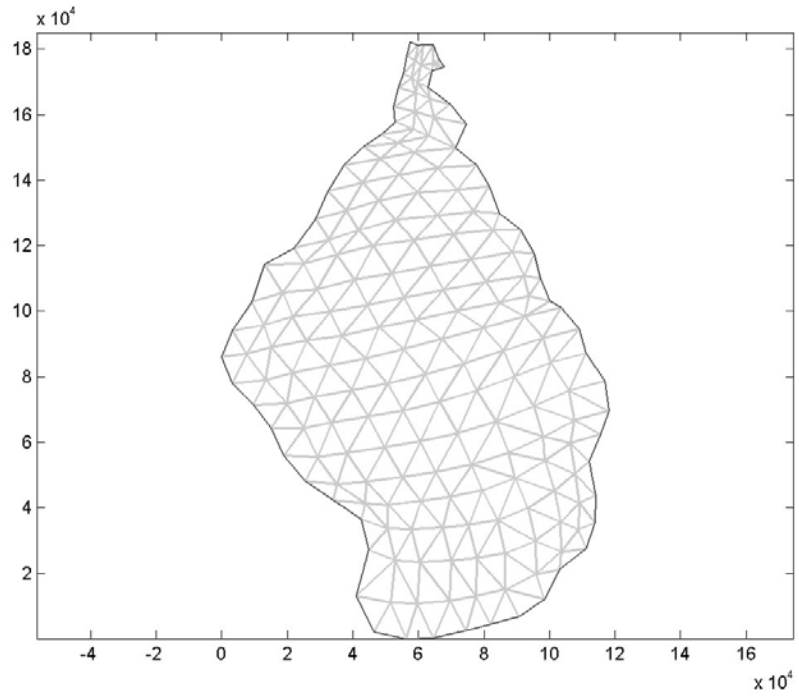
```
[mar2_in,mar2_out]=chomp_grid(mar2,p);
```

6. Add element geometry info

```
mar2_in=belint(mar2_in);  
mar2_in=el_areas(mar2_in);
```

```
figure  
drawelems(mar2_in)  
plotbnd(mar2_in)  
axeq
```

```
mar2_in.name='lake_mcbo2';  
writegrid(mar2_in,'lake_mcbo2')
```



**Figure 32.** The result of the chopping of the *mcbo* mesh. This mesh is the '*lake\_mcbo*' mesh.

## APPENDIX A: BatTri 2D FE GRID GENERATOR

### *Version 7.2.01*

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](mailto: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".

### 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 ShowMe. You can compile and install this in a Unix or Linux X environment to achieve a faster grid display performance. Otherwise, Matlab is used for displaying meshes.
- The most recent version of the OPNML toolbox can be downloaded from [http://www.opnml.unc.edu/OPNML\\_Matlab](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, including example directories, in it. The last thing that needs to be done is to add the directory location of the BatTri to your Matlab path.

Once Triangle, the OPNML toolbox and BatTri are correctly installed, the user should define the location of the Triangle mesh generator and the ShowMe display package (if using Unix or Linux) explicitly. This is done by changing the triangle\_path and showme\_path variables in generate\_mesh.m routine of the BatTri distribution. The path to the ShowMe display package should only be defined if it is compiled and installed in Unix or Linux. Otherwise, comment this line with a %.

## 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. If necessary, all this information can be created from an ordered coastline point data with the use of the editing options of BatTri (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 three 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.
```

## RUNNING BatTri

The Matlab command for running the mesh generator is:

```
generate_mesh(x,y,z,e)
```

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

BatTri will then displays the defined Triangle and ShowMe (if compiled, installed and defined in generate\_mesh.m) paths. In the case the ShowMe path is defined, BatTri will also ask you how the mesh plotting will be handled during the current session. Enter either 1 to use ShowMe or 0 to use Matlab. Remember that ShowMe is a faster option for grid plotting.

You will then be prompted to enter the filename of the final grid ('finalgrid') that will be generated in the BatTri session. This filename will be added 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  $[C_1, C_2, C_3, \dots, C_N]$  for multiple contours or  $[C]$  for a single contour, where  $C$  and  $C_1, \dots, C_N$  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 contours on the screen. If "plotting the locations of the bathymetric data points" option is also chosen later on, the user can actually build the domain boundary by zooming in and adding nodes and segments later on in the mesh editing process. The bathymetric database limits should also define the domain boundary in this case.

- 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. Do not forget to include the apostrophes.

- 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 in apostrophes, 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.

## MESH EDITING

Mesh editing menu consists of 13 options:

- 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 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 exits from the mesh editing menu and saves the changes to the "finalgrid\_triangle.#.poly" file.

## 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<sup>2</sup>, no elements whose area is larger than 50,000 m<sup>2</sup> 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.

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). Then 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. Entering 1 proceeds to the mesh refinement step.

## 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 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 options are supported:

- **$h / [\text{grad}(h)*A]$  (Option 0)** : This makes a colored patch of  $h/[\text{grad}(h)*A]$  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  $h/[\text{grad}(h)*A]$  constraint in the next refinement, you can use the figure colorbar under the plot to select the value for the parameter alpha, so that  $h/[\text{grad}(h)*A] \geq \alpha$ . 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  $h/[\text{grad}(h)*A]$ .
- **$1 / [\text{grad}(h)*A]$  (Option 2)** : Same as Option 0 but for  $1/[\text{grad}(h)*A]$ , instead of  $h/[\text{grad}(h)*A]$ .
- **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<sub>1</sub>, L<sub>2</sub>, and L<sub>3</sub> are the length of the sides. Equilateral triangles have q = 1. Elements with a low quality number can cause numerical problems.
- **Quit (Option 9)** : This quits the diagnostic plotting menu and proceeds to the mesh refinement section.

## 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  $[-\text{Inf}, -0.5]$ . During the next refinement, a constant area constraint can be applied to  $[-.5, \text{Inf}]$ .

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.

- ***h/grad(h) Refinement (Option 0)*** : This scheme uses the following formula to relate the maximum element area to the average element depth and change in depth:

$$h/[(\text{grad}(h))^{\alpha}] \geq A,$$

where  $h$  is the absolute value of the average element depth,  $\text{grad}(h)$  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 1)*** : 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 2)*** : This scheme relates the maximum element area to the change in bathymetry according to:

$$1/[(\text{grad}(h))^{\alpha}] \geq A.$$

Here,  $\text{grad}(h)$  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 3)*** : 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.

- ***Tidal Wavelength to Grid Size Refinement (Option 4)*** : This scheme is based on the Courant condition, which requires a higher grid resolution in shallower areas than deeper water zones because of the slower wave celerities experienced in shallow water. The maximum area that an element can have is calculated using the following formula:

$$[(g \cdot h \cdot T^2)/R^2] \geq A.$$

Here,  $g$  is the gravitational acceleration in  $\text{m}/\text{sec}^2$ ,  $h$  is the average depth of an element in meters,  $T$  is the tidal period of interest in seconds (44714.16 sec for  $M_2$ ),  $R$  is the tidal wavelength to grid size ratio set by the user and  $A$  is the maximum area that an element can have in the final grid. An element minimum inside angle constraint may be needed to top the Courant condition to smooth the grid by creating elements with better aspect ratios.

- ***Constant Maximum Area Refinement (Option 5)*** : This imposes a constant maximum element area constraint on all the elements of the domain, without using the bathymetry.

- ***delta(h)/h Refinement (Option 6)*** : This scheme first checks to see if elements of the current grid meet the

$$\text{delta}(h)/h \leq \alpha$$

criteria. Here,  $\text{delta}(h)/h$  is defined as:

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

where  $h_{\text{max}}$  is the depth of the deepest node on the element and  $h_{\text{min}}$  is the depth of the shallowest. If  $\text{delta}(h)/h$  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) 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).

## DIAGNOSTIC PLOTTING OF THE REFINED GRID

As in Diagnostic Plotting of the Preliminary Grid.

## 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 the Cuthill-McKee algorithm, 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).

## POSSIBLE FUTURE IMPROVEMENTS

The following bullets in no particular order summarize possible future improvements to BatTri that we may implement depending on the reactions that we receive from users:

- Capability to input a scanned geo-referenced map picture (jpeg) to digitize coastline and/or bathymetry for scratch building;
- Capability to move bathymetric data points and to change depth attributes manually;
- Bathymetry based refinement by zone;
- "On the fly" zone specification during refinement (i.e. re-define zones as you refine);
- Automatic identification of islands;
- "Bathymetry approximation error" based constraints;
- triangle2fe\_struct.mex, fe\_struct2triangle.mex (to speed up I.O.);
- scattered\_contour.mex (speed up contouring).

## ACKNOWLEDGEMENTS

We would like to thank B. Blanton (UNC), C. Denham (USGS), D. Fugate (VIMS), T. Gross (NOAA), D.R. Lynch (Dartmouth College), J. Manning (NMFS) and J. Veeremony (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](mailto:ata.bilgili@dartmouth.edu)) and Keston Smith ([ksmith@whsun1.wh.who.edu](mailto:ksmith@whsun1.wh.who.edu)). Please report all errors or comments to Ata Bilgili.