Naval Research Laboratory

Stennis Space Center, MS 39529-5004

NRL/MR/7322--08-9083

MeshGUI: A Mesh Generation and Editing Toolset for the ADCIRC Model

CHERYL ANN BLAIN
Ocean Dynamics and Prediction Branch
Oceanography Division

ROBERT S. LINZELL
Planning Systems, Inc.
Slidell, Louisiana

T. CHRIS MASSEY
Ocean Dynamics and Prediction Branch
Oceanography Division

February 8, 2008

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
08-02-2008	Memorandum Report	

4. TITLE AND SUBTITLE

MeshGUI: A Mesh Generation and Editing Toolset for the ADCIRC Model

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
0602435N

6. AUTHOR(S)

Cheryl Ann Blain, Robert Linzell,* and T. Chris Massey

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER
73-6801-07-5

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
Oceanography Division
Stennis Space Center, MS 39529-5004

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/MR/7322--08-9083

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
800 N. Quincy St.
Arlington, VA 22217-5660

10. SPONSOR / MONITOR'S ACRONYM(S)

ONR

11. SPONSOR / MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

*Planning Systems Inc., MSAAP Building 9121, Slidell, LA 70458

14. ABSTRACT

The User's Manual for the Matlab-based MeshGUI Graphical User Interface (GUI) containing the software programs *meshcreate.m* and *meshedit.m*. describes the software, including its functionality and usage. The *meshcreate* and *meshedit* tools provide a relatively automated and interactive means, respectively, for obtaining good quality finite element meshes that directly interface with the ADvanced CIRCulation Model (ADCIRC) through the *fort. 14* file. The resulting mesh, defined by a coastline boundary that inscribes connected linear triangles, discretizes a geographic region of interest for which the ADCIRC model is to be applied. The goals of the GUI-based *meshcreate* and *meshedit* tools are to provide some flexibility to the experienced user for designing or modifying a mesh while at the same time ensuring a quality mesh will result after only minimal input from the novice user.

15. SUBJECT TERMS

ADCIRC model	Mesh generation	*Fort.14*
Finite element mesh	MeshGUI	Mesh editing

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
					Cheryl Ann Blain
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**	UL	52	**19b. TELEPHONE NUMBER** *(include area code)*
Unclassified	Unclassified	Unclassified			(228) 688-5450

TABLE OF CONTENTS

1. Description

a. Overview

The Meshcreate and Meshedit Tools provide a relatively automated and interactive means, respectively, for obtaining good quality finite element meshes that directly interface with the ADvanced CIRCulation Model (ADCIRC). The resulting mesh, defined by a coastline boundary that inscribes connected linear triangles, discretizes a geographic region of interest for which the ADCIRC model is to be applied. The goals of the Graphical User Interface (GUI) based Meshcreate and Meshedit Tools are to provide some flexibility to the experienced user for designing or modifying a mesh while at the same time ensuring a quality mesh will result after only minimal input from the novice user.

The Matlab program *meshcreate* requires two external data files. Both of these files retain rather generic formats to enhance usability of the software and facilitate interfacing with unknown and varied data sources. The first data file is a boundary file that describes the geographic limits of the mesh to be created. Often the boundary file represents a coastline (the land/sea interface) and includes segments across open ocean waters in order to close the mesh boundary. The boundary file can however be constructed to include overland areas as long as supporting topographic information is available. The boundary file can be created using an auxiliary Matlab utility, *makecoast*, or derived from software such as the Surface Modeling System (SMS). The second required data file is a bathymetry file that contains water depths or land elevations that span the entire region enclosed by the boundary file. This bathymetry file provides the basis for the grid creation and refinement by *meshcreate*. Note that the mesh file created by *meshcreate* includes only the nodal coordinate list and the element incidence list. It does not include information the ADCIRC model expects to describe boundary types. The format written by *meshcreate* is detailed in Appendix Ic, Grid File Format.

The Matlab program, *meshedit*, requires only one external data file. This required data file is the ASCII text finite element grid file to be edited. This grid file must already exist, and can be the output of the companion *meshcreate* software or another mesh creation software package. The program *meshedit* also can accept an finite element grid file that is merely a background mesh of triangulated bathymetry. This type of file has the same format as the required input grid file, and is an optional output of the *meshcreate* package. It often is advantageous to modify a mesh, and then sub-sample bathymetry from the background mesh, interpolating it onto the modified mesh so as to retain fidelity of the original bathymetric data. This operation is described in Section 2c. All input data file formats required by *meshcreate* and *meshedit* are detailed in Appendix I.

b. Methodology

The *meshcreate* software package follows these basic steps when generating a new mesh from bathymetry data and boundary information:

1. A rectangular mesh is created to encompass the domain defined by the boundary file with nodes spaced some initial distance apart; the initial distance is in meters, and is set by the user in the GUI.
2. Additional nodes are added to the grid based on the bathymetry data values; this process is called refinement, and the number of refinements is set by the user in the GUI.

Manuscript approved October 5, 2007.

3. Once the rectangular mesh is refined, the boundary information is used to "cut" the final mesh from the refined rectangular mesh.
4. The boundary quality is then checked, and duplicate nodes are removed. The final boundary will have a resolution that matches that of the refined mesh.

The *meshedit* software package follows these basic steps when the user desires to modify an existing mesh:

1. A mesh file specified by the user is loaded into the GUI and displayed in the GUI plot panel.
2. Modifications are selected and enacted by the user from the GUI. Generally, modifications are performed one-at-a-time over the entire mesh or for a user-delineated region of the mesh. A "best practices" script also can be selected which performs several mesh quality checks and subsequent modifications in sequence over the entire mesh.
3. The user then saves the modified mesh to a new file, either after one or more modifications or after all modifications is performed. It is recommended that the user save an updated version of the mesh at regular intervals to ensure that the final results are of the best possible quality and that intermediate steps can be retrieved if the results of an operation are unsatisfactory (as of this writing, there is no "Undo" function).

c. Software Prerequisites

MATLAB® and the *MeshGUI* software must be installed on the same platform. The *MeshGUI* software has been used extensively with Linux versions 7.1.0.183 (R14) and 7.4.0.336 (R2007a). The GUI software may not be compatible with versions older than 6. Although no UNIX®- or Linux-specific software was developed for the *MeshGUI* package, there may be third-party software in the Matlab repository that is not compatible with comparable Windows™ versions.

For software other than MATLAB®, such as Perl or Fortran programs, the respective Perl interpreter or Fortran compiler must be installed. Additionally, any Fortran programs must be compiled for the computer system on which it is to be executed. An existing bathymetry data and boundary information files for the region of interest must be present in the current working directory. Format descriptions for the required data sources is provided in Appendix I.

d. Software Components

The components of the *MeshGUI* software is composed of several primary MATLAB® modules described below and in Figure 1:

1. *meshgui.m* – a window-based user interface offering three actions to: Create a new mesh, Edit an existing mesh, or Cancel.
2. *meshcreate.m* – the main MeshCreate GUI that provides user interaction to enter required bathymetry data and boundary information file names, set the initial mesh creation parameters, and start or cancel mesh creation.
3. *meshcreate.fig* – the MATLAB® binary ".fig" file that contains the GUI graphics and associated behind-the-scenes software.
4. *Meshcreate.html* – an HTML help file providing information describing GUI usage and mesh creation options.

5. *meshedit.m* – the main MeshEdit GUI that provides user interaction to enter the required edit grid file name, select and perform mesh editing tasks, and display the mesh.
6. *meshedit.fig* – the MATLAB® binary ".fig" file that contains the GUI graphics and associated behind-the-scenes software.
7. *Meshedit.html* – an HTML help file providing information describing GUI usage and mesh editing options.

Routines external to MeshGUI but discussed in the Appendix II and III.

8. *bath2grd.m* – a MATLAB® program that interpolates bathymetry data (in the MeshGUI format) onto an ADCIRC grid.
9. *nnodes_vs_time.m f* – a MATLAB® program to evaluate a mesh's computational requirements (in terms of the number of CPUs based on IBM-P4 hardware).
10. *bnd_extr.f* – a standalone Fortran program that creates the boundary type information required at the tail of an ADCIRC *fort.14* grid file. The program supports the extraction of boundary nodes from an ADCIRC grid file for elevation-specified and no flux boundaries.
11. *XmGREDIT* – a third-party, GUI-based program for editing an ADCIRC mesh file with capabilities for creating ADCIRC model compatible boundary information for the *fort.14* grid file.
12. *cst2bdy.pl* – a Perl script that converts shoreline data from The Surface Modeling Software (SMS) into a format compatible with the MeshGUI software.
13. *makecoast.m* – an interactive MATLAB® program for creating an ordered, closed coastal outline, in a format required by the MeshGUI software, from segmented coastline data.

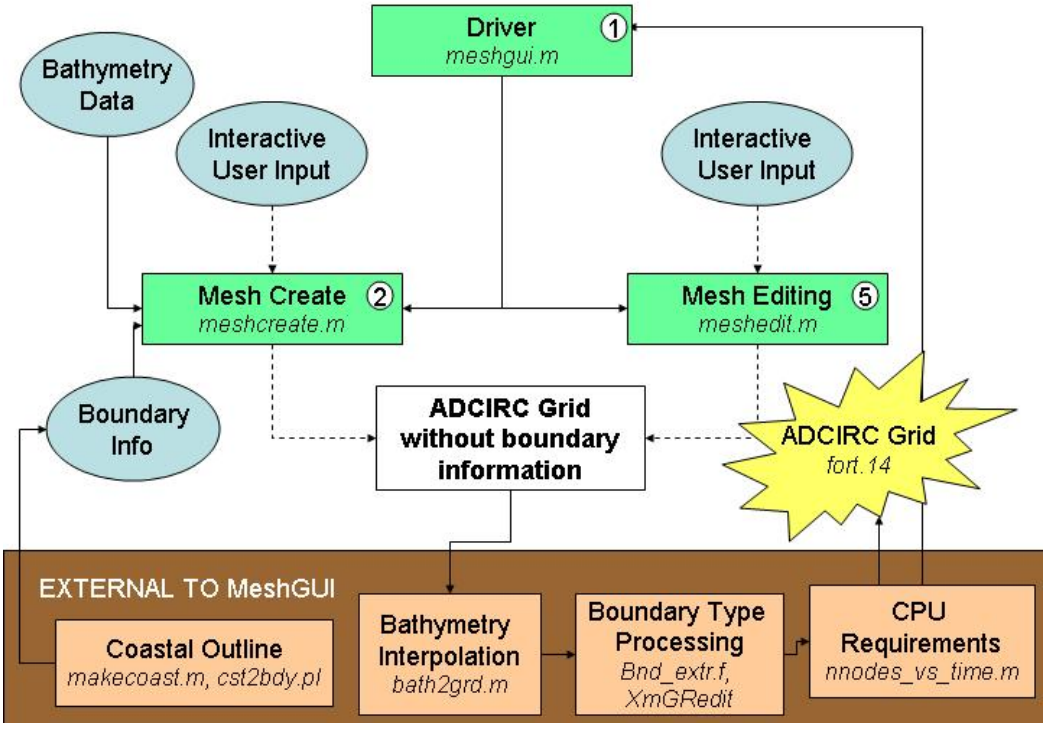


Figure 1. A schematic flow chart depicting the data flow and relationships between software components comprising the MeshGUI tools.

The simple, top-level GUI, *meshgui*, is shown in Figure 2.

Figure 2. Top-level GUI, *meshgui*, for starting the *meshcreate* program.

This small interface prompts the user for which mesh operation to choose. It is invoked by entering "*meshgui*" at the MATLAB® command prompt (without the quotation marks), or by navigating in the Current Directory panel of the main MATLAB® window, and selecting the directory entry, right clicking, and selecting the Run menu item. The user should select the **Create** button to execute *meshcreate*. The user should select the **Edit** button to execute *meshedit*.

Upon selection of the **Create** button, the *meshcreate* GUI, shown in Figure 3, appears upon program startup. A recent GUI upgrade incorporates a status window, shown near the bottom of Figure 2 that displays status, warning, and error messages issued by the various Matlab routines during execution.

Upon selection of the **Edit** button, *meshedit* issues a small, pop-up window as shown in Figure 4. This window notes that the user should select the **Load Edit Mesh** button after the *meshedit* GUI starts. This window disappears after three seconds or if the user clicks the **OK** button or closes it with the **Close Window** button in the top right corner. After the window closes, *meshedit* begins.

The *meshedit* GUI is shown in Figure 5 as it initially appears upon program startup. A recent software upgrade incorporates a status window shown in the bottom left corner of the GUI that displays status, warning, and error messages issued by the various MATLAB® routines during execution. As seen in Figure 5, the plot panel is the white box that makes up most of the *meshedit* GUI. This is where the mesh and its associated features are displayed.

Mesh modification operations are listed in the scrolling box in the upper left corner of the GUI. Plot options are grouped in the box in the center of the left side of the GUI; the user can select one or more of these options to display various features of the mesh. The status message box is in the bottom left corner. Various file loading and saving buttons are on the left and bottom sides. Controls for changing the view of the mesh are on the right side of the GUI. The user can zoom into any region of the mesh with the **Zoom** button, and can move the zoomed in view using the **Pan** button with the mouse pointer. The user also can select a region of the mesh using the **Region** button and clicking with mouse. The **1:1** button resets the view to the entire mesh.

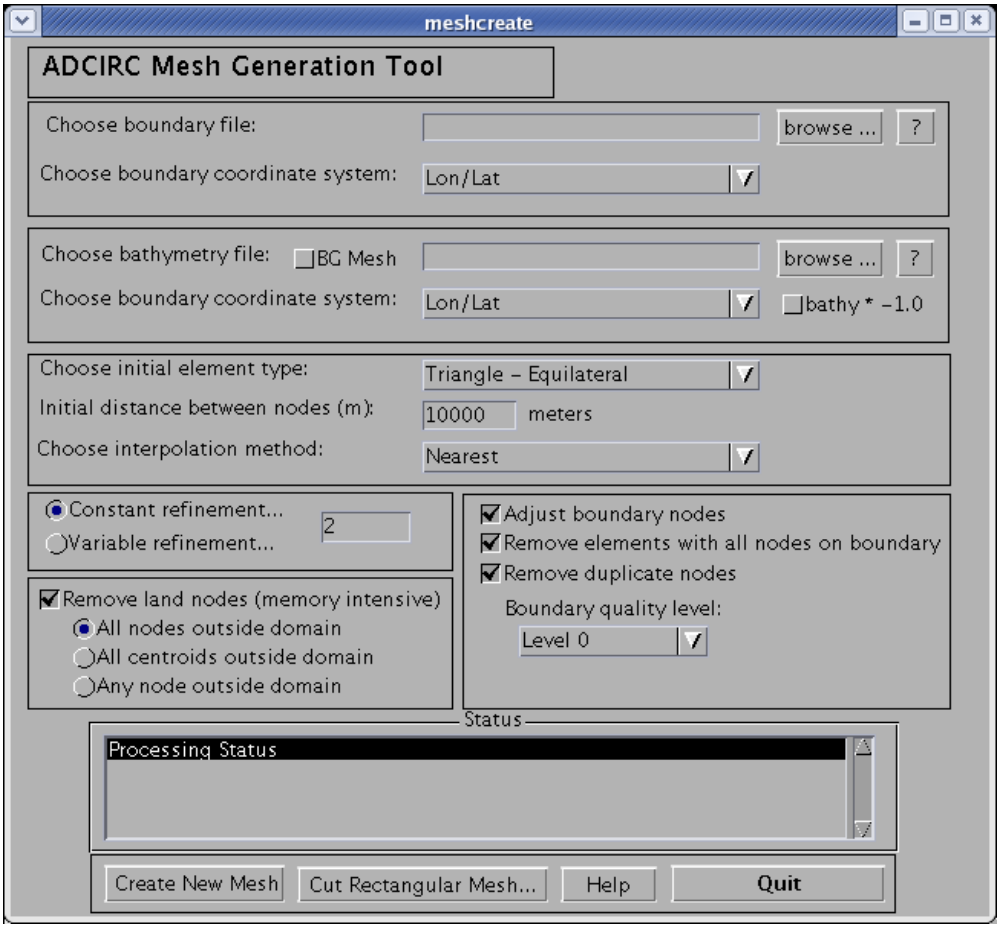


Figure 3. *Meshcreate* GUI interface at startup.

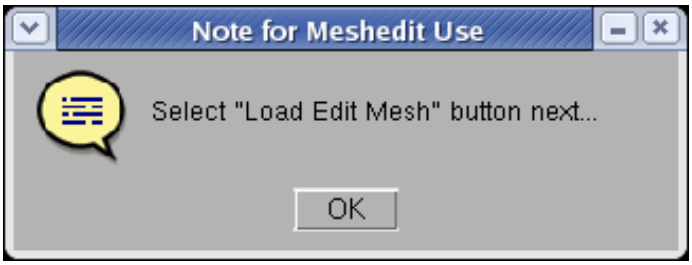


Figure 4. Pop-up window with a brief note on *meshedit* usage.

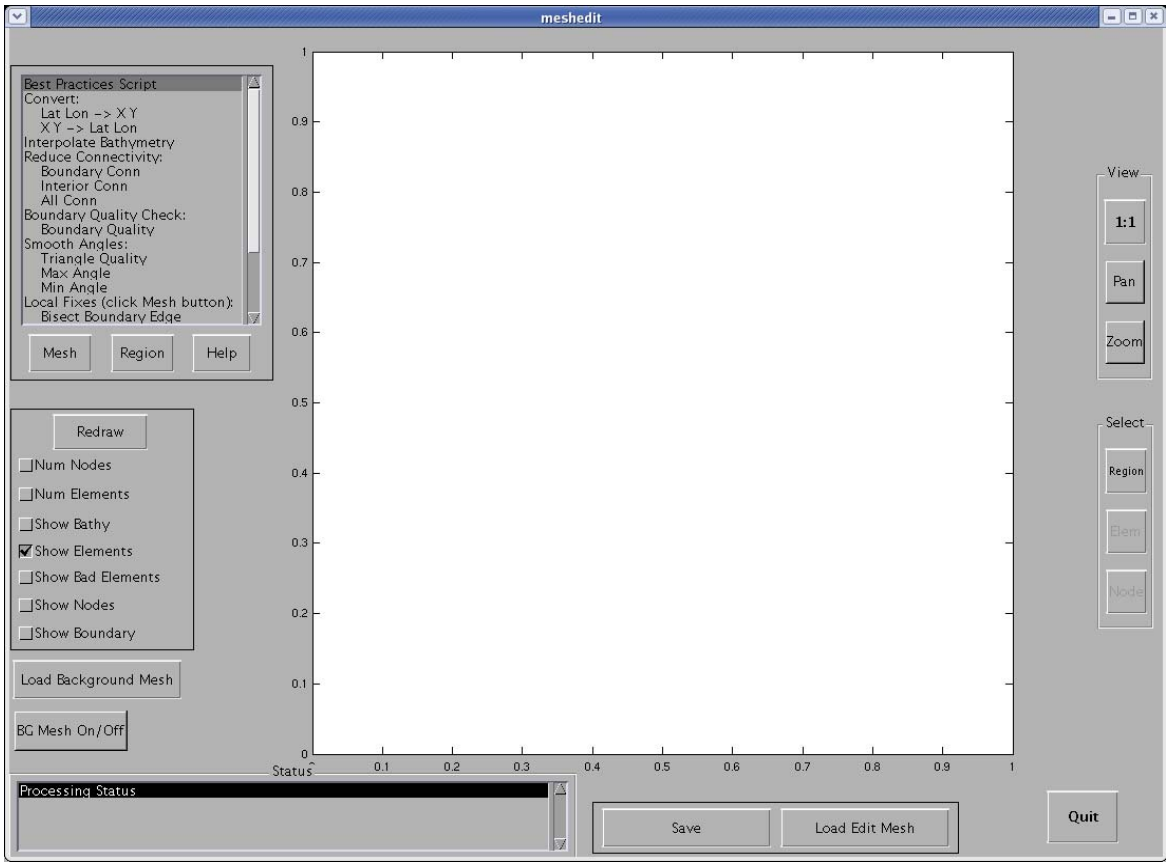


Figure 5. *Meshedit* GUI interface at startup.

Most of the MeshGUI elements, or "widgets," have a tool tip (also known as "bubble help"), which is a small, pop-up window containing a brief description of the widget that appears near the mouse pointer when the user places it on or near an item of interest. Figure 6 shows an example of a tool tip. This example shows a tool tip for *meshcreate* under the "Initial distance between nodes (m)" option. Figure 7 shows a *meshedit* tool tip for the **Mesh** under the button. The tool tip describes the function of that button.

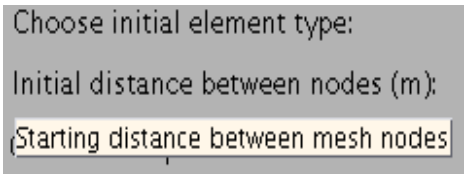


Figure 6. Example of a *meshcreate* tool tip.

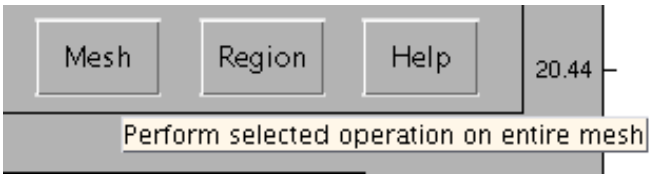


Figure 7. Example of a *meshedit* tool tip.

The included HTML file is displayed in a help browser when the user clicks the Help button near the bottom of the *mechcreate* GUI (Figure 2). This help facility is another recently added upgrade. Figure 7 shows a portion of the *meshcreate* HTML help file displayed in the browser.

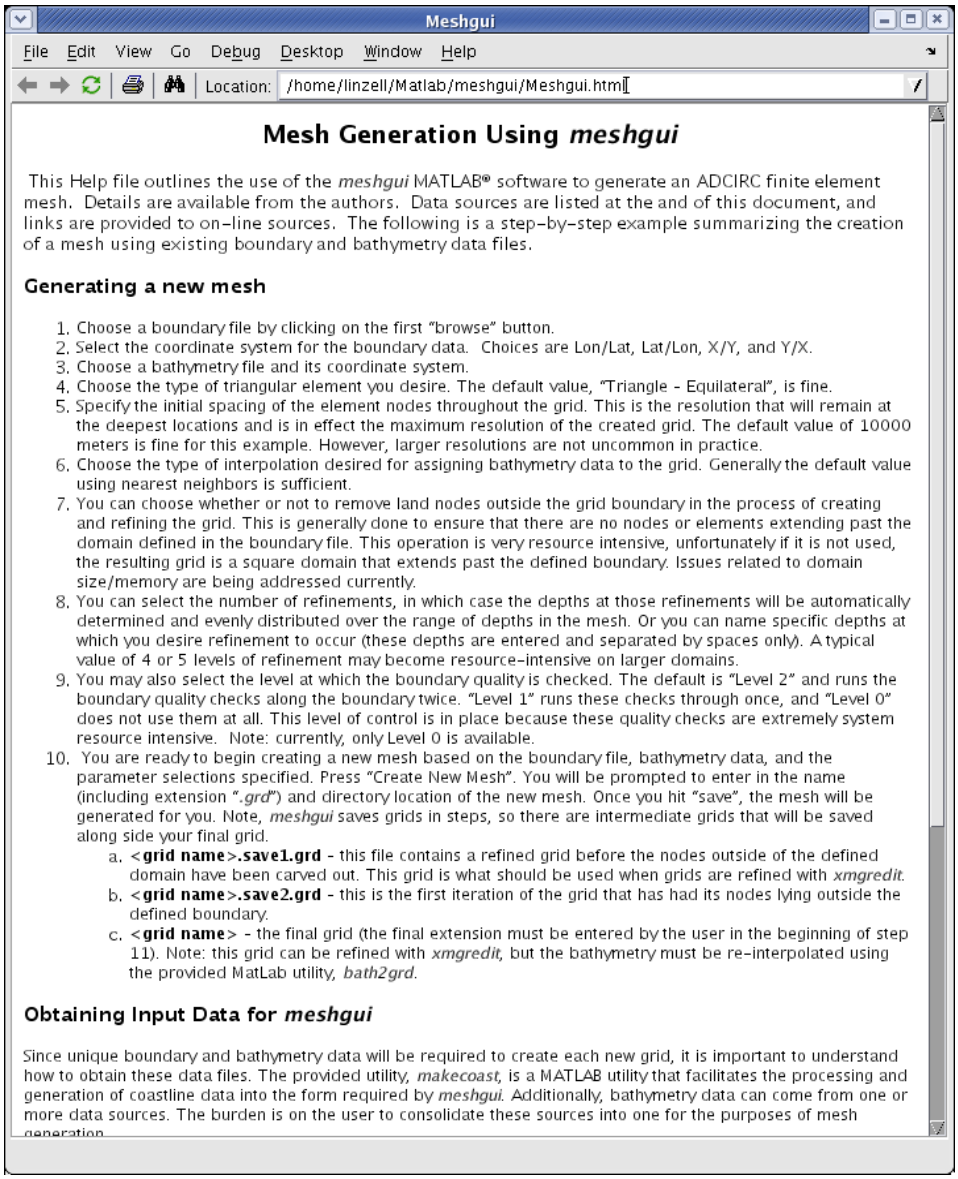


Figure 8. The Help browser displaying the *meshcreate* HTML help file.

Similarly, the HTML file included with the *meshedit* software is displayed in a help browser when the user clicks the Help button on the bottom of the Operations list box located at the top left of the *meshedit* GUI (Figure 4). Figure 9 shows a portion of the *meshedit* HTML help file displayed in the browser.

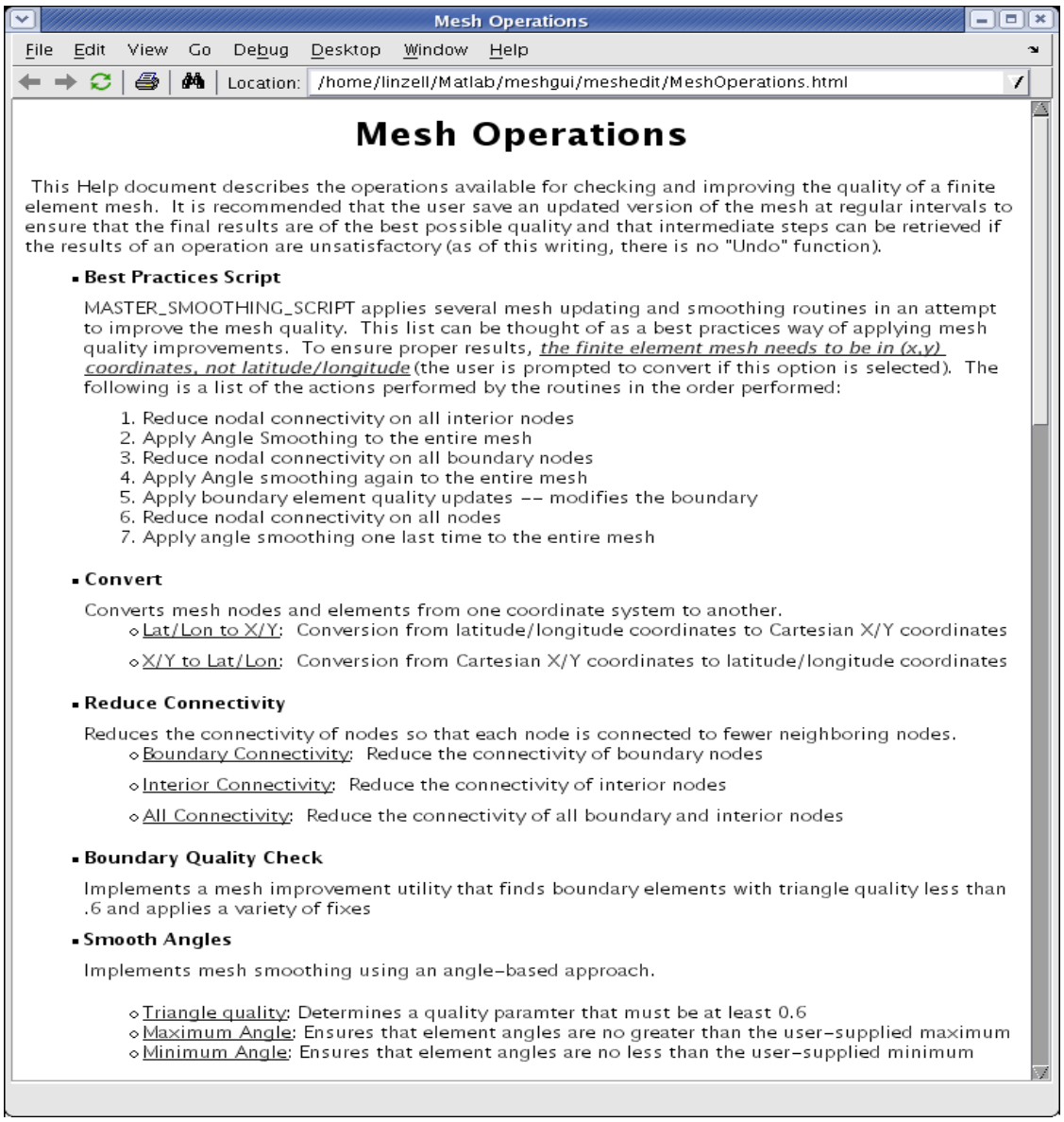


Figure 9. The Help browser displaying the *meshedit* HTML help file.

e. Software Installation

The *MeshGUI* utility is located in the Matlab repository in the meshgui directory. The directory has the following contents:

- Contents.m
- *meshgui.m*
- (directory) documentation/
  - ADCIRC_MANUAL_MeshGUI_FINAL_09-25-07.doc
- (directory) meshcreate/
  - *adjustbe.m*

    - *bdyll2xy.m*
    - *bndqual.m*
    - *bwidth.m*
    - *cleanstruct.m*
    - *en_del.m*
    - *initbndqual.m*
    - *initmesh.m*
    - *inpoly.m*
    - *interpds.m*
    - *loadcoast2.m*
    - *loadfile.m*
    - *makestruct.m*
    - *meshcreate.fig*
    - *Meshcreate.html*
    - *meshcreate.m*
    - *postprocess.m*
    - *refinemesh.m*
    - *resolvebndy.m*
    - *rldecode.m*
    - *rlencode.m*
    - *rmdupnodes.m*
    - *sortrows.m*
- (directory) meshedit/
    - *do_bisect_bndry_edge.m*
    - *do_line_swap.m*
    - *do_merge2bndry_elems.m*
    - *do_move_node.m*
    - *do_split2.m*
    - *get2elems_runproc.m*
    - *get_bad_elems.m*
    - *meshcheck.m*
    - *meshedit.fig*
    - *meshedit.m*
    - *MeshOperations.html*
    - *sortrows.m*

Locations of routines external to MeshGUI:

- (MATLAB directory) grid_util/
    - *bath2grd.m*
    - *nnodes_vs_time.m*
- (MATLAB directory) gencoast/
    - *makecoast.m*
- (Fortran directory) ADCIRC_Utilities/Boundary_Processing/Physical
    - *bnd_extr.f*
    - *cst2bdy.pl*
- *XmGREDIT:* http:// www.ccalmr.ogi.edu/CORIE/software, ACE

Step 1: MATLAB® Path Configuration

Once the user has extracted the archived MATLAB® software to its final location, it must be made available to MATLAB®. The top level `/Matlab` directory (e.g., `/home/[username]/Matlab` on a UNIX®/Linux system, where "`[username]`" should be replaced with the user's actual login name) and the subdirectories under it must be added to the MATLAB® path. There are several methods for modifying the path; details can be found in the MATLAB® documentation available through the Help browser or in the command window.

The NRL MATLAB® software ships with a sample startup script named, *startup.sample.m* in the top level `/Matlab` directory. The user should copy this script to his/her personal MATLAB® directory (e.g., `/home/[username]/MyMatlab`) and rename it to *startup.m*. If there is an existing file of this name, then the new version can be appended to the existing one. Alternatively, this file can be copied to the MATLAB system startup directory; the MATLAB software documentation provides details on this option. This file should then be edited with a text editor to change any NRL-specific paths to the user's system-specific paths. Two additional methods for invoking MATLAB® with the customized startup.m script, involving a user-defined alias and a user-customized shell script are described here.

(OPTIONAL) Step 2: MATLAB® Alias Configuration

Once the new *startup.m* file has been suitably edited, then the user's login environment can be modified (if it has not already been modified) to invoke MATLAB® using this file. One method is to configure an alias in the user's login scripts (e.g., *.profile* or *.cshrc* in the home directory). An alias allows the user to issue a short command that contains one or more longer commands. Two examples are as follows:

```
example% alias matlab 'cd $HOME/MyMatlab; /path/to/Matlab &' [for csh/tcsh]
example% alias matlab='cd $HOME/MyMatlab; /path/to/Matlab &' [for sh/bash]
```

in which the first example defines an alias for the C shell or TC shell, and the second for the Bourne shell or bash shell. The actual path to the MATLAB® binary executable program would be substituted for "`/path/to/Matlab`" in the above examples. The user would merely enter the command "matlab" at a command prompt to change to the personal MATLAB® directory in which *startup.m* resides, and start the program.

(OPTIONAL) Step 3: MATLAB® Shell Script

Another method for invoking MATLAB® with a custom *startup.m* script would be to create a shell script that is either invoked by name from the command line, or activated by a button from the user's console window. To employ this method, the user would create a script using a text editor, and place it in a directory that is in the user's path. An example for the C shell or TC shell, named "*runmat.csh*" and residing in the user's home directory, is as follows:

```
example% cat ~/runmat.csh
#!/bin/csh
cd $HOME/MyMatlab
/path/to/Matlab –desktop –r startup &
```

This example works like the earlier shown alias examples. The user can then enter the script name at a command prompt, or create a console button that invokes the script by clicking on the button. Details of button creation vary with the windowing environment; the user should refer to the software documentation or consult a local system administrator.

2. Mesh Generation: Using *meshcreate*

a. Getting Started

The user interface for *meshcreate* is aimed at simplifying application of the mesh generation tool. The following is a step-by-step procedure for the creation of a mesh using the *meshcreate* GUI shown in Figure 2 and user-supplied bathymetry data and boundary information files:

1) Start MATLAB®, ensuring that *meshcreate* is set up properly (see Installation, Sect. 1e)

2) Choose a boundary information file by clicking on the first "browse" button.

3) Select the coordinate system that reflects the form of the boundary information data, e.g. [longitude, latitude], [latitude, longitude], [X,Y], or [Y,X].

4) Choose the name of the bathymetry file and its coordinate system, similar to the choices for the boundary information file. If necessary, select the check box ("bathy * -1.0") to reverse the sign of bathymetry values that are negative (the ADCIRC model expects positive depth values). Select the "BG Mesh" check box if the bathymetry file is a previously created background mesh file from the *meshcreate* software. This option allows the user to omit the internal step of triangulating the bathymetry if one is using the same bathymetry data to create multiple grids (e.g., one may be creating several smaller grids within a larger geographic region).

5) Choose the type of triangular element you desire. The default value, "Triangle – Equilateral", is the best choice.

6) Specify the desired initial spacing of the element nodes throughout the grid. This is the resolution that will remain at the deepest locations in the domain and is in effect the maximum resolution of the created grid. Specification of the initial resolution and the number of refinements (see 9)) dictate the total number of nodes in the final mesh and the finest resolution achieved in the mesh generation process.

7) Choose the type of interpolation desired for assigning bathymetry data to the grid nodes. Generally the default method of nearest neighbors is sufficient.

8) Choose whether or not to remove land nodes outside the grid boundary in the process of creating and refining the grid. This is generally done to ensure that nodes and elements do not extend beyond the domain defined by the original boundary information file. Note that this operation is very resource intensive. In some circumstances it may be desirable not to perform this step if the user wishes to retain the refined rectangular grid encompassing the domain defined by the boundary information file.

9) Select the number of refinements, for which case the depths at those refinements will be automatically determined and evenly distributed over the range of depths in the mesh. Or specific depths may be named at which refinement will occur (these depths are entered and separated by spaces only). A typical value of 4 or 5 levels of refinement may become resource-intensive on larger domains. Specification of the initial resolution (see 6) above) and the number of refinements dictate the total number of nodes in the final mesh and the finest resolution achieved in the mesh generation process.

10) Select the level at which the boundary quality is checked. Currently, "Level 0" is the only active option. The "Level 1 and "Level 2" boundary quality checks are under development.

11) Press "Create New Mesh" begin mesh generation using the supplied boundary information file and bathymetry data, and the selected parameter specifications. You will be prompted to enter in the name (including extension ".grd") and directory location of the new mesh to be created. Once you select "save", the mesh will be generated.

Note, *meshcreate* saves the generation process as a series of grids reflecting the different step of the mesh generation process. The saved grids include:

a. **<grid name>.save1.grd** – The refined grid prior to the removal of nodes and elements outside of the supplied boundary information file.
b. **<grid name>.save2.grd** – The initial grid prior to refinement, uncut by the boundary data, with the coarsest nodal spacing as specified by the user.
c. **<grid name.grd>** – The final grid (the final extension must be entered by the user in the beginning of step 11). Note: this grid can be refined with *xmgredit*, but the bathymetry must be re-interpolated using the provided Matlab utility, *bath2grd*.

After the bathymetry data are triangulated (during Step 11), the user is prompted for whether to save the new bathymetry mesh as a background mesh for later use. Figure 10 shows the prompt.

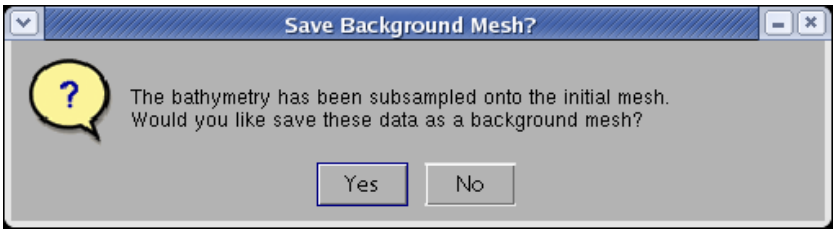


Figure 10. The *meshcreate* window pop-up for retention of the Background Mesh.

If the user chooses to save the mesh, a file selection dialog appears in which the user can enter a new file name or select an existing file. The background mesh data are then written to the file, and a status message listing the output file is printed in the status window. If the user chooses not to save the mesh, only a status message is issued. Program execution continues once the user interaction is complete.

After the initial coarse mesh has been created, the bathymetry data sub-sampled onto it, and any user-specified refinements have been performed, the user is prompted for whether or not to run the Master Smoothing Script. The prompt is shown in Figure 11.

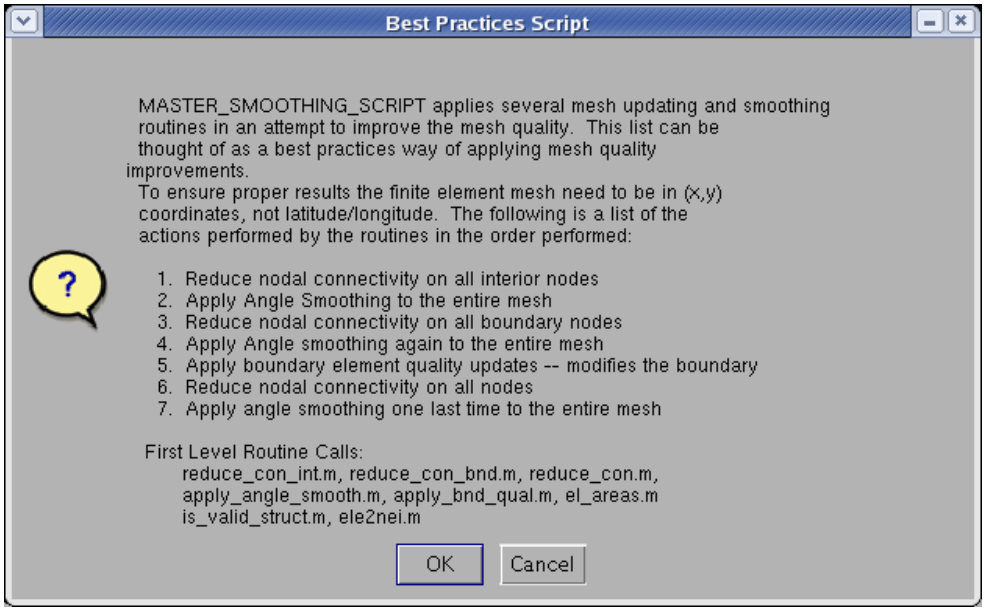


Figure 11. *Meshcreate* user prompt for execution of the Master Smoothing Script.

As seen in Figure 11, the prompt contains a description of the operations to be performed by the Master Smoothing Script. If the user selects the **OK** button, the script operations are performed on the mesh and the status of the operations is printed in the status window. If the user chooses not to run the script, the program continues execution without running the script. Execution of the master smoothing script is highly recommended to obtain the best quality mesh.

Once all mesh generation operations have been completed, the user is prompted for whether to open the newly created mesh in the mesh editing GUI, *meshedit*. This prompt is shown in Figure 12.

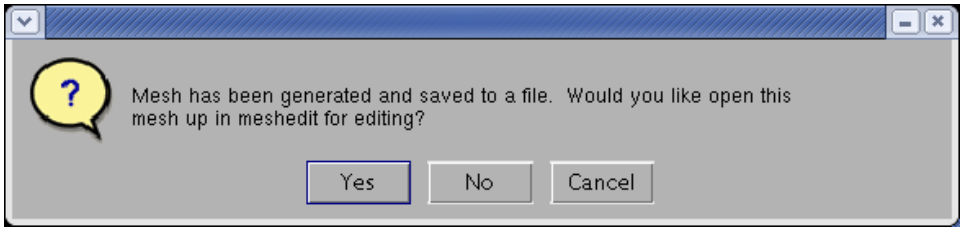


Figure 12. *Meshcreate* user prompt for launching *meshedit* if visualization, diagnostics or editing of the newly created mesh is desired.

If the user selects the **Yes** button, the *meshcreate* GUI is closed, and the *meshedit* GUI is opened with the newly created mesh displayed. Otherwise, *meshcreate* waits for further user input. The options are to exit the program, to create another new mesh, or to open an existing (rectangular) mesh for which boundary cutting operations is desired (e.g. further refinements were made to an existing rectangular mesh and that mesh is now imported into *meshcreate* for removal of elements and nodes outside the supplied boundary information file (Figure 3).

b. Localized Grid Refinement

The refinement by depth criteria employed by *meshcreate* has limitations in adequately capturing the geometry of deep, narrow channels. As such it may be necessary to refine such regions external to *meshcreate* using *meshedit*. The *Split-4* refinement process is described in Section 3b (*Split-4* for a region). Once the refinement process is complete, the refined mesh (typically a rectangular grid encompassing the original boundary data) can be imported into *meshcreate* and a final mesh obtained using the following procedure:

1) Start up meshcreate in MATLAB®.
2) Once in *meshcreate*, select the original boundary information and bathymetry data files used for the creation of the original grid.
3) Press "Refine Existing Mesh" and select the modified grid saved from the *meshedit* refinement.
4) Provide a name to which the final refined mesh will be saved.
5) Repeat as necessary to achieve the desired resolution and geometric representations in the final grid.

Experience has shown that grids that have been refined AND remain uncut by the boundary data (i.e., retain their initial rectangular bounding box) are prone to require computational resources that may exceed local capacity. If this approach proves to be too computationally intensive for the available computer resources, consider modifying the final grid (i.e., the non-rectangular, "cut" grid) and re-interpolating the bathymetry using *bath2grd* (see Section 2c below).

c. Bathymetry Interpolation: Using *bath2grd*

To re-interpolate the original bathymetry data to the finished grid or interpolate new bathymetry data to an existing grid, the Matlab utility, *bath2grd,* is needed. This interpolation utility is run within MATLAB® and requires an existing grid and a bathymetry data file that has the same format required for *meshcreate* – see Appendix I). The utility *bath2grd* (located in the *grid_util* directory of the Matlab repository) is invoked using the command:

```
>>bath2grd('oldfile','newfile','bath.dat');
```

(* *note the single quotes around the file names)*

where `oldfile` is the name of existing grid,
`newfile` is the name of the new grid file that is created, and
`bath.dat` is name of the bathymetry data file that is to be interpolated to `oldfile`.

A recent upgrade to the program *bath2grd* added options for a coordinate transformation, converting from spherical (degrees latitude/longitude) to Cartesian (meters X/Y) coordinates for more accurate interpolation. The syntax is as follows:

```
>>bath2grd('oldfile','newfile','bath.dat',ICS,rlat,rlon);
```

where `oldfile, newfile,` and `bath.dat` are as above defined, and the new options are defined as:
`ICS` is a flag indicating the coordinate system (1 for Cartesian, or 2 for spherical coordinates),
`rlat` is the latitude in degrees of the centroid location for the CPP projection, and
`rlon` is the longitude in degrees of the centroid location for CPP projection;

and CPP is the Carte Parallelo-grammatique projection. The values of `rlat` and `rlon` can be obtained using the interactive Matlab program, *delt_interact*, with the existing grid file (`oldfile`). This program should be executed prior to executing *bath2grd*, unless the user already has known values for `rlat` and `rlon`.

d. Mesh Computational Requirements: Using *nnodes_vs_time*

To evaluate the computational requirements of a created mesh the Matlab utility, *nnodes_vs_time,* is needed. This calculator is run within MATLAB® and requires estimates for the number of nodes in the grid, the time step size, the simulation length, and the total wall clock time allowable for operational execution. The interactive MATLAB® program returns an estimate of the necessary number of CPUs required to complete the simulation as specified by the user-input parameters, the number of nodes per processor which is a reflection of the communication overhead (too few nodes/processor will result in large communication costs) and a graph of the CPU time in hours vs. the number of CPUs. The utility *nnodes_vs_time* (located in the *grid_util* directory of the Matlab repository) is invoked using the command:

```
>>nnodes_vs_time
```

The following interactive questions require user input:

```
Enter the number of nodes in the mesh:
Enter the time step size for the run in seconds:
Enter the run length in days:
Enter the total wall clock time you can allow in hours:
```

The calculator then produces the following evaluation of the mesh parameters supplied:

```
To finish this run in # hours, will require # CPUs.
The number of nodes per processor will be about #.
```

If the number of nodes per processor falls below 1000, the following warning is issued:

```
Caution:
This many processors may negatively impact the actual timing
due to the small number of mesh nodes per processor increasing
the communication overhead.
```

A suggested number of processors is made and the estimated run time is recorded:

```
A reasonable number of processors could be #
which would require # wall clock hours.
```

The computational time estimates computed within the *nnodes_vs_time* utility are empirically derived by curve fitting actual ADCIRC model simulation times for a particular computer platform knowing that ADCIRC model scaling is nearly linear on most machines. Obviously, as computer processor units become faster and communication speeds improve, the empirical relationship assumed may become outdated.

e. Extracting Boundary Type Information: Using *bnd_extr.f*

The final step required to obtain a properly formatted ADCIRC grid file (*fort.14*) (see adcirc.org/documentation/fort_14.html) is the extraction of boundary nodes and the assignment of boundary types to these nodes. This information is appended to the nodal coordinate and element incidence list in the grid file created by MeshGUI. The Fortran utility, *bnd_extr.f* (located in the *ADCIRC_Utilities/Boundary_Processing/Physical/* directory of the Fortran repository) is available to accomplish this task. The *bnd_extr.f* program must be compiled using a Fortran compiler for the system on which the program will be executed. Required input for *bnd_extr* is a grid file without boundary information, such as that saved by MeshGUI; *the name of this input grid file must be* "neibound.in". Boundary node extraction begins by typing the name of the binary executable:

```
>>bnd_extr
```

The *bnd_extr* code issues the following interactive questions that require user input:

```
Enter the number of open boundaries
```

The user must provide the total number of disconnected open ocean boundaries that will require elevation specified forcing. The beginning and ending nodes of each open boundary segment is required in advance and input at the next program prompt:

```
Enter the beginning and ending node on each o.b.
Open boundaries may be entered in any order,
But beg/end nodes for each must be CCW.
Open boundary #
```

The open boundary node numbers are entered on the screen one at a time followed by a [return]. Successful completion of the code results in the statement:

```
FORTRAN STOP
```

The program *bnd_extr* generates two boundary files

gredit.bnd – a boundary file in a format accepted by the XmGREDIT software package, and
adcirc.bnd – a boundary file that can be appended to the MeshGUI grid file to create a valid ADCIRC grid file, *fort.14, e.g.*

```
cat neibound.in adcirc.bnd > fort.14
```

For this Fortran routine, "open boundaries" refers to elevation-specified open boundaries only. Boundary nodes not contained within the open boundary node segments input into the code

(above) are assumed to be no-flux type boundary nodes (land). The extraction and boundary type assignment of non-zero flux specified boundaries is not supported. Such boundaries would occur at the entrance of rivers into the domain or at an offshore boundary for which non-zero flux values were to be specified. To properly specify these types of boundaries the boundary information file must be edited according to the specifications outlined in the online ADCIRC model manual, http://www.adcirc.org/documentation/fort_14.html.

3. Mesh Editing: Using *meshedit*

a. Starting *meshedit*

The GUI for *meshedit* is designed to simplify modification and ensure high quality elements within a created mesh. To begin, the user clicks the **Load Edit Mesh** button located near the lower right corner of the GUI (Figure 5). A file selection dialog appears as shown in Figure 13.

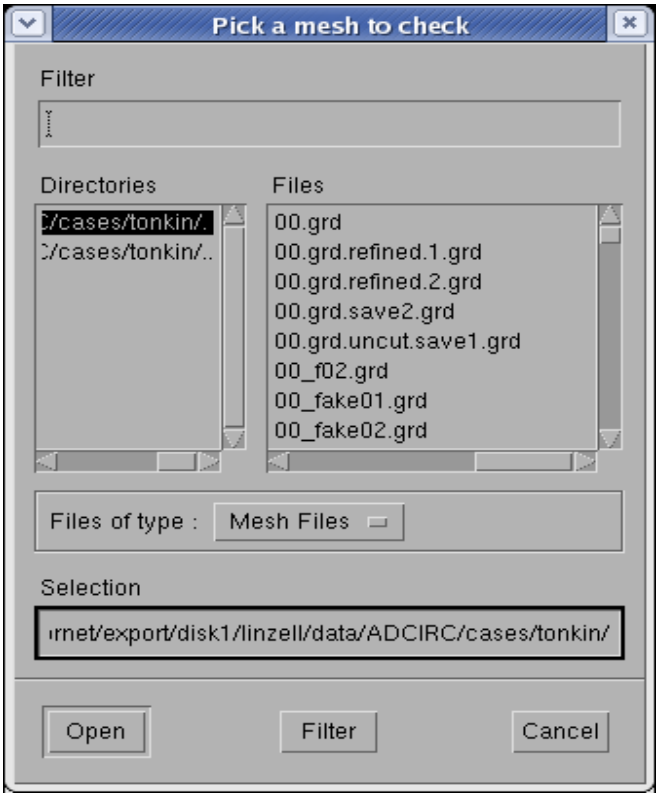


Figure 13. File selection dialog for opening a mesh file within *meshedit*.

The user double-clicks on the name of the mesh file to be edited, or enters the file name manually, and the file is opened. The mesh is displayed in the plot area of the GUI as shown in Figure 14.

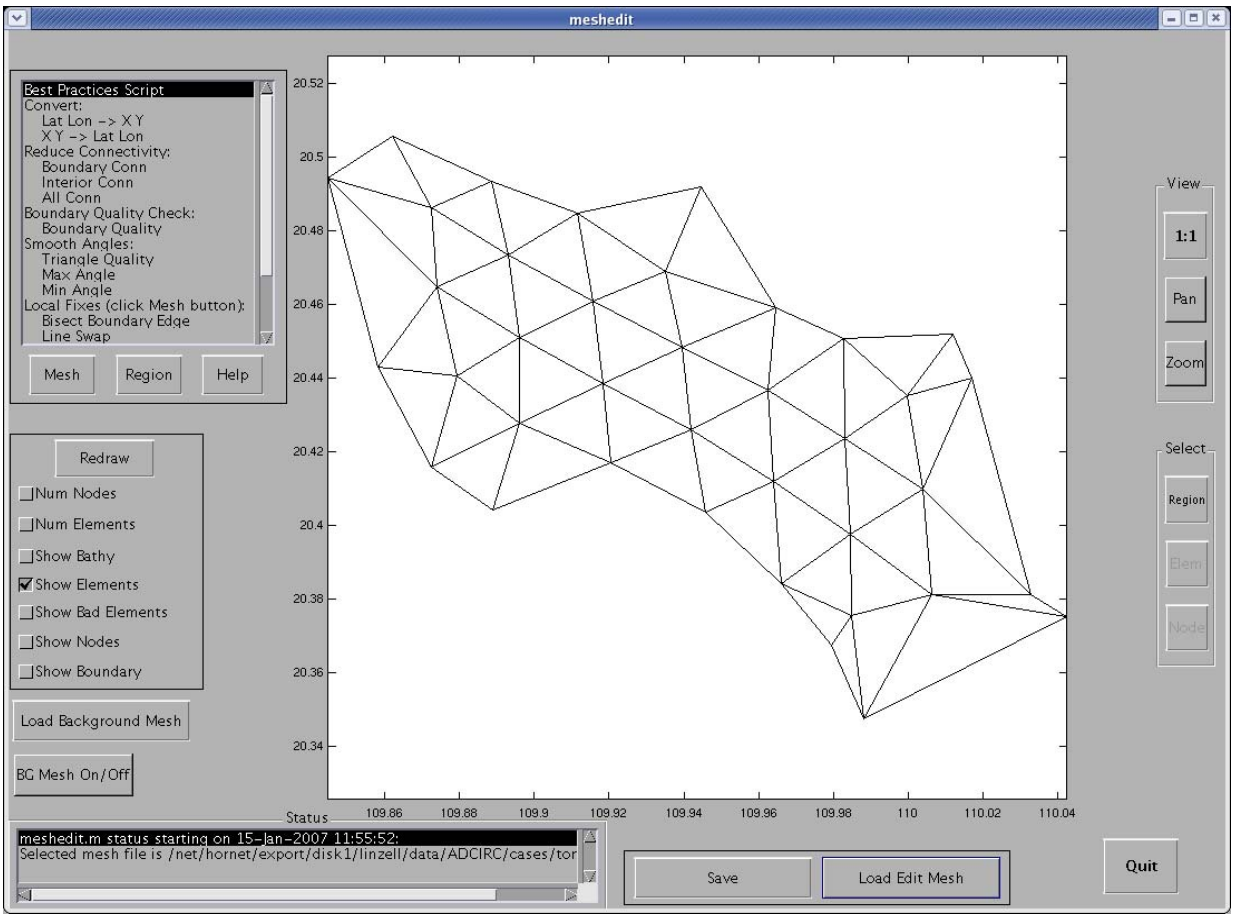


Figure 14. *Meshedit* GUI displaying the mesh selected for editing.

As shown in Figure 14, the mesh to be edited, hereinafter referred to as the "edit mesh," is presented in the plot area with only the mesh elements displayed. The user can then add other mesh attributes to the plot or proceed with the editing process. Details of the available mesh editing operations (described in Section 3b) and the mesh display attributes (described in Section 3c) are presented in the order of their appearance within the GUI.

b. Mesh Operations

The available mesh operations are located in the upper left scrolling panel in the *meshedit* GUI (Figure 5).

- ***Best Practices Script***

Selection of this option executes a Matlab script called the *master_smoothing_script* that applies several mesh adjustment and smoothing routines in order to improve overall mesh quality. The script is a "best practices" approach to applying available mesh quality improvements. To appropriately apply these mesh manipulations, the mesh *must be in Cartesian (X/Y) coordinates* instead of spherical (latitude/longitude) coordinates. If the user selects this option, the program issues a prompt for whether to apply the coordinate transformation, as shown in Figure 15.

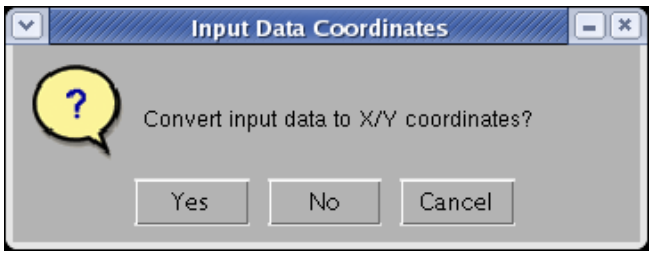


Figure 15. The *meshedit* mesh operations user prompt from the "Best Practices" selection for converting the mesh to X/Y Cartesian coordinates.

If the user chooses not to convert by clicking the **No** or **Cancel** buttons, the script continues without converting the coordinates. *This should be done only if the edit mesh currently loaded is already in X/Y Cartesian coordinates.* If the user clicks the **Yes** button to convert the mesh coordinates, the program issues a prompt for the coordinates of the centroid of the CPP transformation. This prompt is shown in Figure 16.

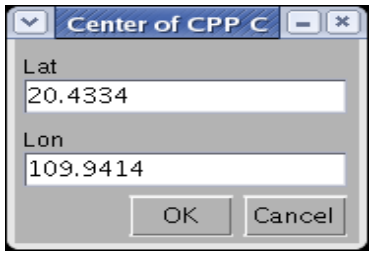


Figure 16. The *meshedit* mesh operations user prompt from the "Best Practices" selection for the coordinates of the centroid of the CPP coordinate transformation.

The mean latitude and longitude of the mesh are displayed in the prompt, and the user can modify them prior to clicking the **OK** button. If the **Cancel** button is clicked, the script continues, but will *not* convert the coordinates to and X/Y frame.

After the coordinate conversion prompts are issued, the user is prompted to enter a minimum value for the triangle quality ("triqual") parameter. This prompt is shown in Figure 17.

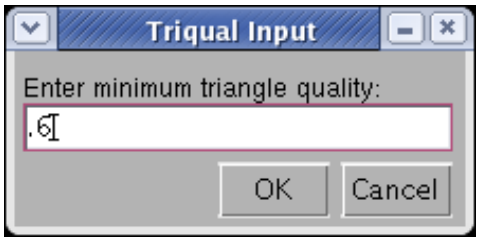


Figure 17. The *meshedit* mesh operations user prompt from the "Best Practices" selection for the triangle quality parameter.

As shown in Figure 17, the default value is 0.6. The default value has been selected as a balance between an ideal goal of equilateral triangles (i.e., triqual = 1.0) and establishing a maximum internal angle threshold of less than 120° for an element. Higher values of the triqual parameter

are more restrictive on the allowable element shape and entail significantly greater computational effort. Triangle quality measures less than 0.6 could result in ill-formed elements that affect stability of the model over the mesh and, ultimately, could degrade the computed solution for that mesh.

Following specification of a triangle quality parameter, the *meshedit* GUI displays a window containing descriptive text that summarizes the "Best Practices" procedures. Figure 18 displays this window.

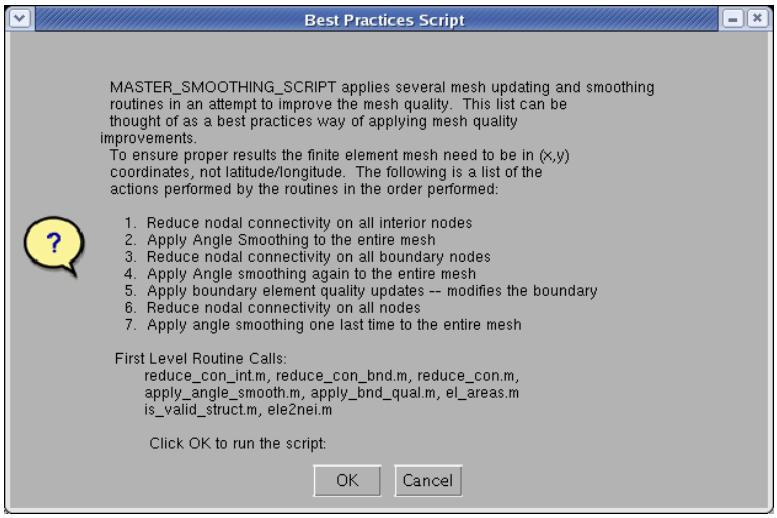


Figure 18. The *meshedit* mesh operations information window from the "Best Practices" selection describing the various mesh quality control procedures.

The "Master Smoothing" procedure in Figure 18 is defined in order as:

- Reduce nodal connectivity on all interior nodes
- Apply angle smoothing to the entire mesh
- Reduce nodal connectivity on all boundary nodes
- Apply angle smoothing again to the entire mesh
- Apply boundary element quality updates, which modifies the boundary
- Reduce nodal connectivity on all nodes
- Apply angle smoothing one last time to the entire mesh

The Matlab functions employed by the script are also listed. The user must select the **OK** button to continue execution of the script. Execution of the script can be canceled by selecting the **Cancel** button. During script execution, status messages are printed to the MATLAB® command window and in the status window of the *meshedit* GUI. A brief summary of any mesh changes are printed at the end of the script run. After the script has completed execution, the user can save the modified mesh, then exit the program or continue processing the mesh.

- ***Convert***

This subset of mesh operations converts mesh nodes and elements from one coordinate system to another. The Lat/Lon→X/Y option converts from latitude/longitude coordinates to Cartesian X/Y coordinates. The X/Y→Lat/Lon option converts from Cartesian X/Y coordinates to latitude/longitude coordinates. As noted for the Best Practices Script option, *all mesh editing operations must be performed in Cartesian coordinates*. The user must ensure that the conversion is performed, if necessary. prior to further mesh operations.

- ***Interpolate Bathymetry***

Interpolate Bathymetry is a newly added option that allows the interpolation of bathymetry data onto a new mesh. The user inputs either an existing background mesh (defined below) or a simple ASCII text "XYZ" format (i.e., three columns containing, respectively, longitude, latitude, and depth) file to provide bathymetry values for interpolation onto the edit mesh. A background mesh is an ASCII text mesh file (whose format is described in Appendix I) containing triangulated XYZ bathymetry data. The triangulation process creates a set of triangle vertices optimally formed from the coordinates of the bathymetric data. Typically a "background mesh" is created from a bathymetric data set while an "edit mesh" is formed using other element formation criteria such as employed in *meshcreate* or equivalent software.

If the user selects this operation and then clicks the **Mesh** button in the Mesh Operations paenl (Figure 3), a prompt (Figure 19) appears requesting the type of bathymetric data to use.

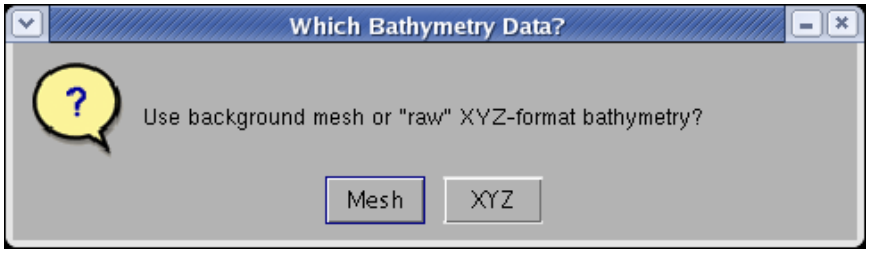


Figure 19. The *meshedit* mesh operations user prompt from the "Interpolate Bathymetry" option requesting selection of the form of the bathymetric data to be used.

If the user selects the **Mesh** button shown in Figure 19, the program checks whether a background mesh already has been loaded. If so, then the user is prompted for whether to use that background mesh or a new one, as shown in Figure 20.

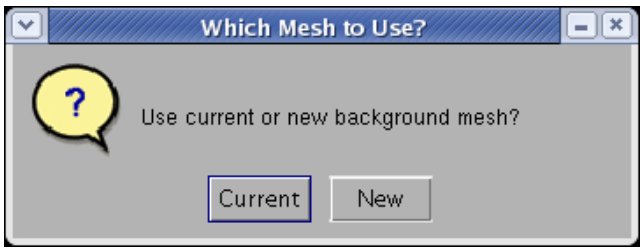


Figure 20. The *meshedit* mesh operations user prompt from the "Interpolate Bathymetry" option requesting selection of a background mesh.

If the user selects the **Current** button as shown in Figure 20, the currently loaded background mesh is used as the bathymetry data source for interpolation. If the user selects the **New** button, a file selection dialog (e.g., Figure 13) is opened for the user to select a different background mesh file. The selected background mesh then served at the bathymetry data source for the interpolation.

Alternatively, if the user selects the **XYZ** button (Figure 19), then a file selection dialog (e.g., Figure 13) is opened for the user to select a bathymetry data file. The user-specified file is loaded into MATLAB®, and the bathymetry is triangulated, as is done during the creation of a "background grid". The user is then prompted for whether to save this newly triangulated bathymetry as a background mesh for later use, as shown in Figure 21.

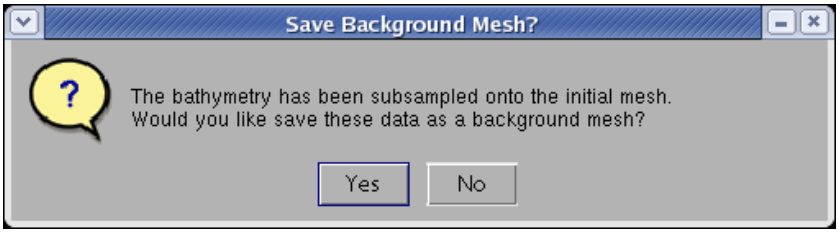


Figure 21. The *meshedit* mesh operations user prompt from the "Interpolate Bathymetry" option requesting whether to save the triangulated bathymetric data as a background mesh.

If the user selects the **Yes** button (Figure 21), then a file save dialog appears as shown in Figure 22 for the user to specify a new file name for the output mesh, and the file is saved.

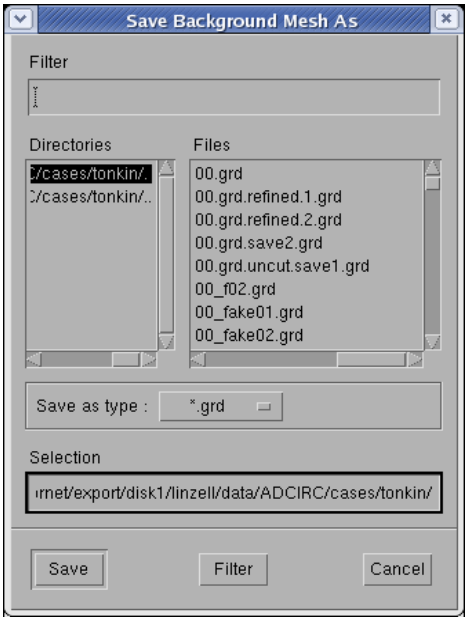


Figure 22. The *meshedit* mesh operations file save dialog for saving a background mesh issued through the "Interpolate Bathymetry" option.

.
As can be seen in Figure 22, the user can double click on an existing file name to overwrite that file, or enter a new file name manually to save the background mesh to a new file. If the user chooses not to save the background mesh, select either the **No** button in the Save Background Mesh prompt (Figure 21) or the **Cancel** button in the file save dialog (Figure 22), and the bathymetric the data are not saved as a background mesh. Once the user interaction is complete, the newly triangulated bathymetric data are interpolated onto the edit mesh.

- ***Reduce Connectivity***

This operation reduces the connectivity of the elements through a single node so that each node is connected to fewer than 7 neighboring elements. Nodal connectivities of twenty or more elements are handled by the software. Perfectly equilateral triangles result in six elements connected to a given node. Three options are available for application of this reduce connectivity feature:

- Boundary Connectivity - reduces the connectivity of boundary nodes
- Interior Connectivity - reduces the connectivity of interior nodes
- All Connectivity - reduces the connectivity of all nodes, boundary and interior

- ***Boundary Quality Check***

The *Boundary Quality Check* mesh operation enforces a triangle quality value of 0.6 or more (see discussion of Figure 17) to all boundary elements and applies a variety of other enhancements to the boundary elements.

- ***Smooth Angles***

A subset of angle smoothing operations implements mesh smoothing using an angle-based approach. Three options are available:

- Triangle Quality - enforces an element quality whose value is 0.6 or higher
- Maximum Angle - ensures that element angles are no greater than the user-supplied maximum
- Minimum Angle - ensures that element angles are no less than the user-supplied minimum

If the user selects the Triangle Quality option, the “triqual” prompt shown in Figure 17 appears. The default minimum value of 0.6 or larger should be used for triangle quality. For the Maximum Angle option, the user is prompted to enter a maximum angle value to replace the default of 115.0°, as shown in Figure 23, or select the **OK** button without changing the value to use the default. It is recommended that the default value be used. Once the desired maximum angle is entered, the software searches all of the mesh elements to determine the angles of each triangle, flags any angles that exceed the maximum, and attempts to modify any element having a flagged angle.

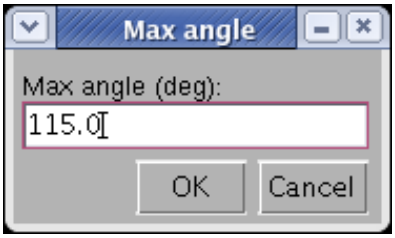


Figure 23. The *meshedit* mesh operations file user prompt for a maximum angle issued through the "Smooth Angles" option.

If the user selects the Minimum Angle option, the program prompts the user to enter the minimum angle value, as shown in Figure 24.

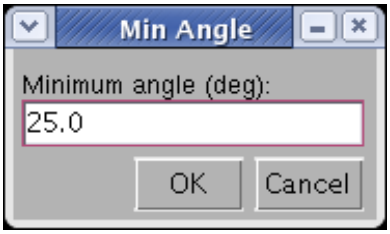


Figure 24. The *meshedit* mesh operations file user prompt for a minimum angle issued through the "Smooth Angles" option.

As for the Maximum Angle option, the user can enter a value to replace the default of 25.0°, or select the **OK** button without changing the value to use the default. It is recommended that the default value be used. Once the desired minimum angle is entered, the software searches all of the mesh elements to determine the angles of each triangle, flags any angles that are less than the minimum, and attempts to modify any element having a flagged angle.

The following mesh operations are implemented element by element, one element at a time and are thus labeled, **Local Fixes**.

- ***Bisect Boundary Edge***

The *Bisect Boundary Edge* mesh operation is performed on individual user-selected boundary elements to reduce the size of large elements or to add single elements where sparse coverage occurs. The result is an increase in resolution along the boundary by forming two elements where one existed through the bisection of one edge of the original element. Once *Bisect Boundary Edge* is selected, a set of crosshairs appears in the plot region of the GUI. The user is instructed (by a message in the status window) to select a boundary element with the mouse. Once the desired boundary element is selected, the software bisects the element's edge that lays on the physical boundary between node numbers NN1 and NN2 (Figure 25). The result is the addition of a new node (NN4) and a new element (E2) to the finite element structure. The area and boundary fields in the *fem_struct* Matlab structure also are updated, and if a nodal neighbor table is present within the Matlab workspace then it is updated as well. The sketch in Figure 25 shows a hypothetical boundary element before (left panel) and after (right panel) the bisect operation.

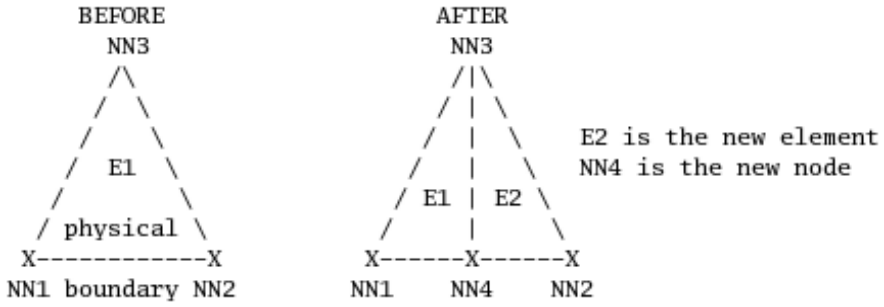


Figure 25. Sketch of the *boundary bisect* operation on a hypothetical boundary element. On the left is the element (E1) before bisection, with the boundary along the bottom edge, while on the right is the bisected element, identifying the new element (E2) and node (NN4).

- ***Line Swap***

The Line Swap option is a mesh refinement tool that swaps the nodal connectivity of an edge shared by two elements. The shared edge between two elements is disconnected and reconnected via the two remaining nodes of the four nodes associated with the two elements. The edge is "swapped". This routine operates on element edge at a time and can be used only if the two elements share an edge. As with the *Bisect Boundary Edge* option, selecting the *Line Swap* option activates a pair of crosshairs with which the user selects the two adjacent elements sharing a common edge that is to be swapped. Figure 26 highlights a region containing two elements that are candidates for a line swap.

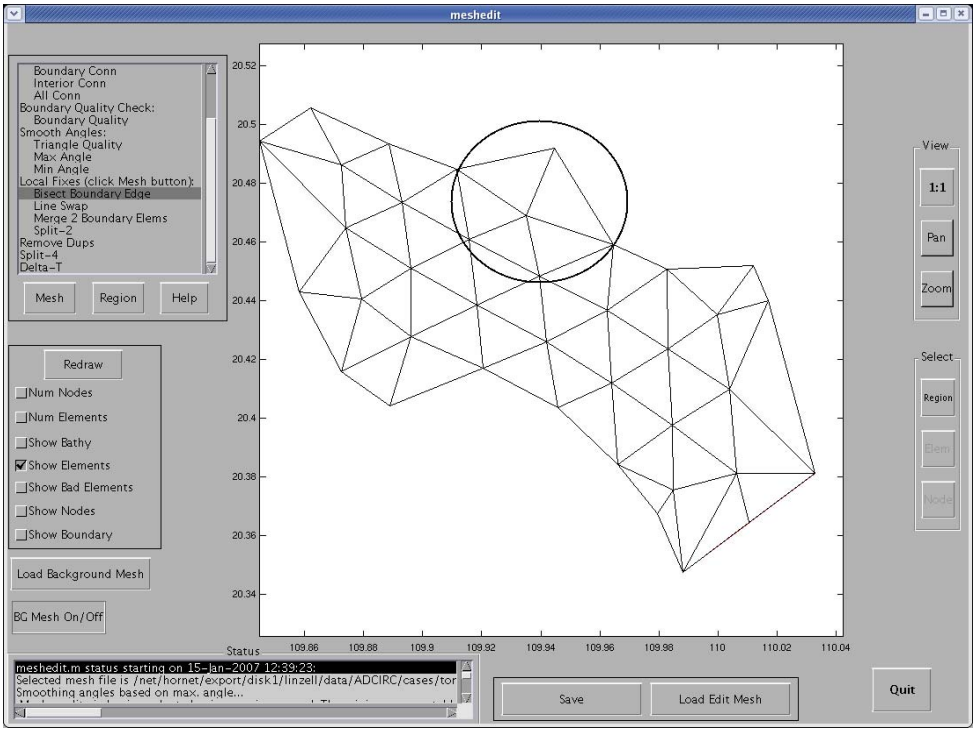


Figure 26. Edit mesh highlighting (circled) a pair of elements that share a common edge to be swapped. The circle was added to emphasize the elements of interest.

The user selects two elements with the crosshairs, and the orientation of their shared edge is modified. The original shared edge is now displayed as a red line while the new edge, perpendicular to the original edge, is displayed as black (Figure 27). This example is for illustration only, since the line swap operation results in a severely skewed element, an undesirable quality inn any mesh.

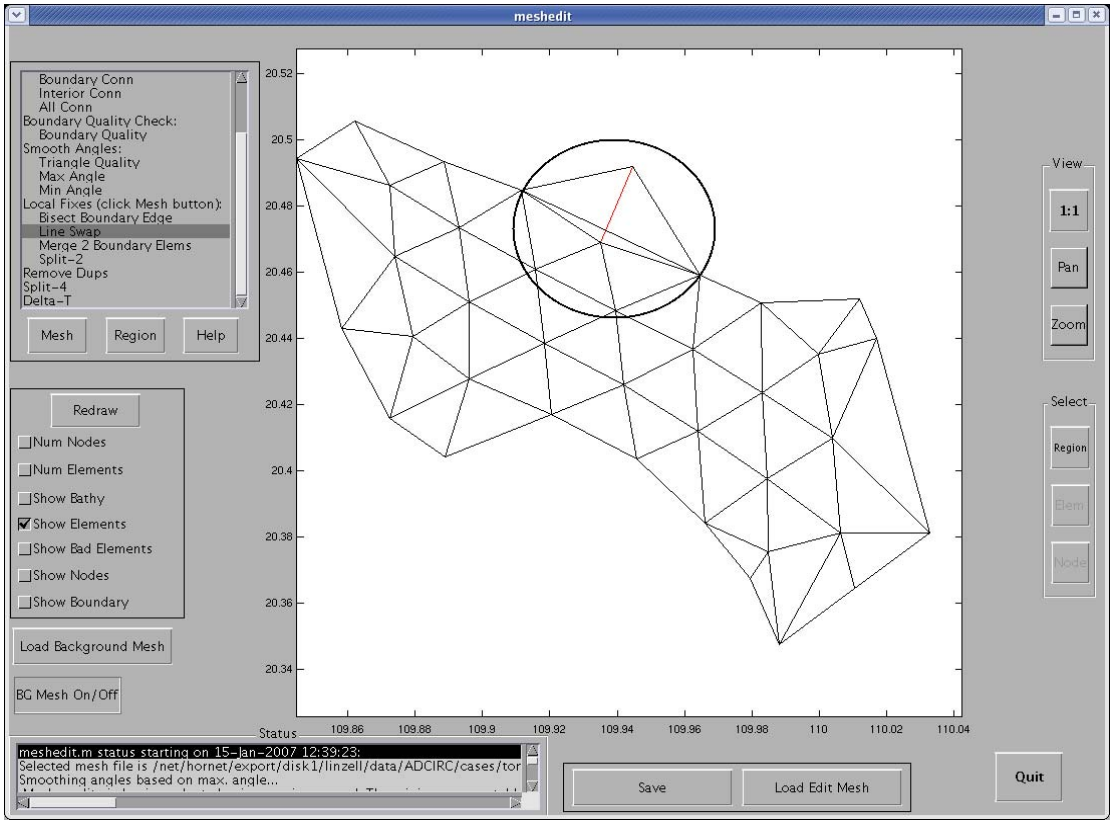


Figure 27. Edit mesh highlighting (circled) a pair of elements having undergone a line swap operation. The new edge orientation is shown in black and the original edge is shown in red. The circle was added to emphasize the elements of interest.

- ***Merge 2 Boundary Elements***

This option merges into a single element two adjacent boundary elements that have a common edge and adjacent edges along boundary. The result is the elimination of one element and one from the finite element structure as well as the removal of one boundary segment. Element areas are updated within the *fem_struct* Matlab structure and, if present, the nodal neighbor list (*.nei) is also updated. Similar to the *Line Swap*, this operation requires the user to select two adjacent elements using the crosshairs. The sketch in Figure 28 illustrates the operation on a pair of hypothetical boundary elements.

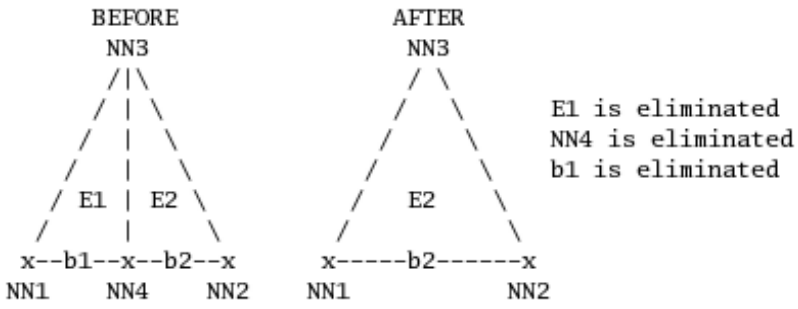


Figure 28. Sketch of the *merge 2 boundary elements* operation on two hypothetical boundary elements. On the left are two adjacent boundary elements (E1 and E2), while on the right is shown the merged boundary element (E2). Identified are the eliminated boundary element (E1), boundary node (NN4), and boundary edge (b1).

The sketch in Figure 28 portrays a pair of adjacent boundary elements before (left hand side) and after (right hand side) merging. One of the elements (E1 in this example) is eliminated, as is one of the boundary nodes (NN4) and boundary edges (b1). This option would be chosen if, for example, a very narrow boundary element was adjacent to a much wider element. The two could be merged to remove the narrow, skewed element.

- ***Split-2***

The *Split-2* operation is a mesh refinement option that splits into four elements the region spanned by two adjacent elements. The *Split-2* proceeds by placing a new node at the midpoint of the shared element edge, then rejoin all of the original nodes to the new node. If the adjacent elements are boundary elements then the new node is placed on the boundary and the element is split using the new boundary node. As described for the *Line Swap*, this operation requires the user to select two adjacent elements using the crosshairs. The sketch in Figure 29 illustrates the operation on hypothetical interior and boundary elements.

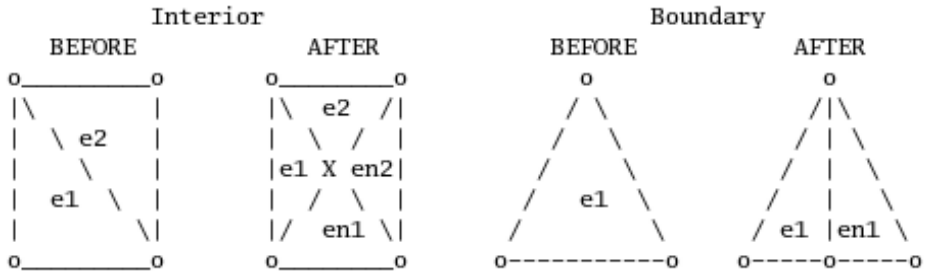


Figure 29. Sketch of the *Split-2* operation on interior (left) and boundary (right) elements. The two interior elements are each split into two elements (left), while a single boundary element is split into two. New nodes also are created at the center of the two elements (left) and in the center of the boundary edge (right).

As depicted by Figure 29, the *Split-2* operation for interior elements (left hand side of the sketch) creates a new element in place of each of the two original ones. For a boundary element, the

Split-2 operation forms two elements in place of the original boundary element. Only one new node is added in each scenario and the Matlab FEM structure is updated accordingly.

- ***Remove Duplicates***

This option is used to remove duplicate elements and nodes from the FEM structure. It operates on the entire mesh.

- ***Split-4***

The *Split-4* operation is a mesh refinement option that splits an individual element into four new elements. The first step splits an element into four elements. This is accomplished by first inserting new nodes at the midpoints of the three edges of the element; and connecting these mid-edge nodes to create a central element. The remaining three elements are formed by connecting the edges of the central element with the vertices of the original triangular element. The final step corrects for "hanging" nodes, nodes that occur at the mid-point of an element edge. The hanging nodes are connected to element vertices that bisect their adjacent element. The *Split-4* operation for a single element ultimately results in an increase from four to ten elements over the same area. The *Split-4* operation can be selected for the entire mesh or for a user-specified region of the mesh. A sketch in Figure 30 demonstrates the successive steps of the *Split-4* procedure on a hypothetical element surrounded by three neighboring elements.

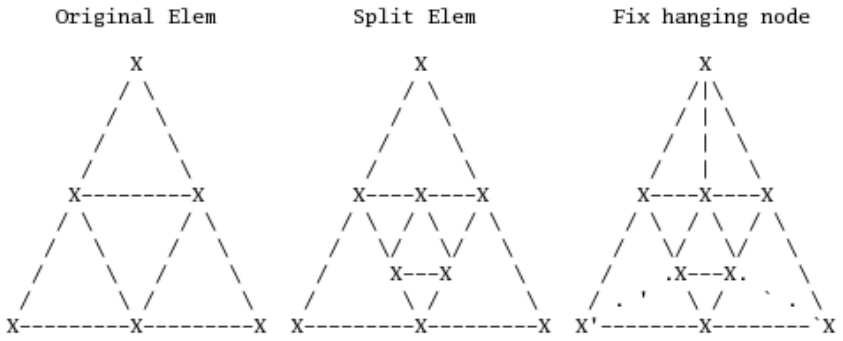


Figure 30. Sketch of successive steps of the *Split-4* proceedure. The four elements on the left are the original elements. The element in the center is refined by the *Split-4* operation (middle) such that four new elements are created. The resulting "hanging" nodes are rectified by bisecting adjacent elements (right) to form six new elements. The elements on the right show the final element configuration.

Figures 31 through 33 show the *Split-4* operation for a user-specified region of the mesh. Figure 31 presents the user prompt issued by the GUI when the **Region** button (e.g., Figure 5) is selected.

Figure 31. The user prompt for selection of the **Region** vertices issued by MeshEdit's Mesh Operations, **Region** button.

After choosing to "proceed", the user mouse selects points that define a polygonal region; the [Enter] key is pressed to complete the selection. A selected region is outlined in red on Figure 32.

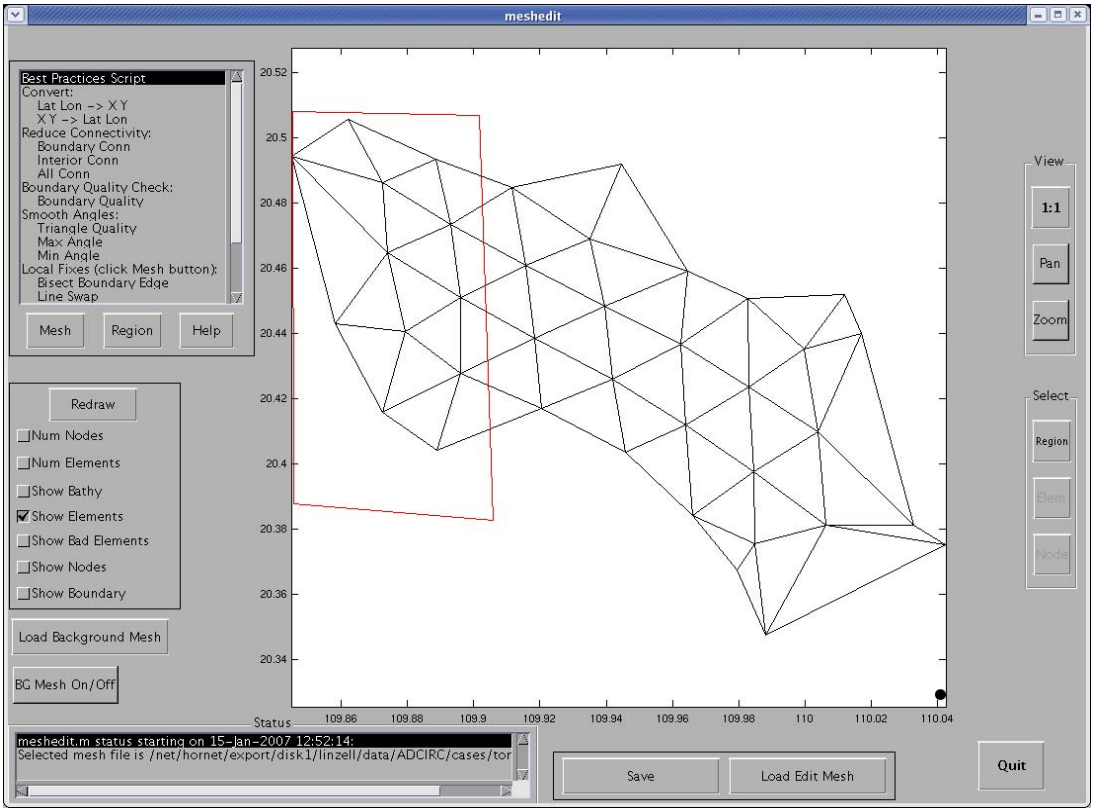


Figure 32. Display of a user-selected polygonal region (red) within which the *Split-4* operation in *meshedit* is applied.

The user then selects the *Split-4* option from the list of Mesh Operations and clicks the **Region** button. At this time, all elements within the region are split (Figure 33). Any elements that cross the boundary of the region also are split.

This operation is useful for adding elements in regions where a higher density is needed to resolve geometric features or increased mesh resolution is desired over the entire mesh. One common application is to import a rectangular, refined mesh created by *meshcreate* prior to cutting by the boundary information and refine areas that are not well-resolved by the

bathymetric depth refinement criteria (e.g., deep, narrow channels or straits) using the *Split-4* operation over a region. The mesh resulting from this application of *Split-4* can be re-imported into *meshcreate* to complete the boundary cut. The result will be a mesh that better resolves the geometric features of interest at the desired resolution.

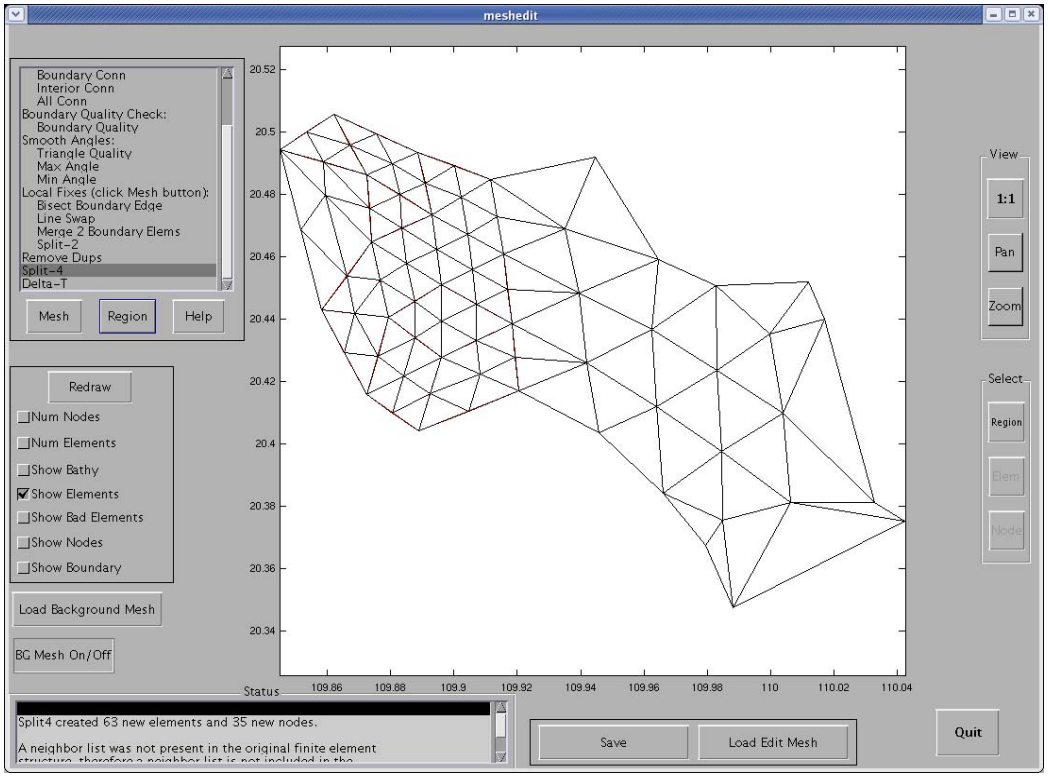


Figure 33. The refined mesh after application of MeshEdit's *Split-4* operation within a user-generated region.

- ***Delta-T***

The final option in the list of available mesh operations in *meshedit* is *Delta-T*. The purpose of the *Delta-T* routine is to provide an estimate of the model time step required to meet a specified Courant-Friedrichs-Lewy (CFL) stability condition. The time step estimate assumes a tidal wave propagation speed equal to *sqrt(gh)* and requires satisfaction of a CFL condition of 0.6 or less over the entire mesh. User interaction for the *Delta-T* operation is performed using the MATLAB® command window shown in Figure 34.

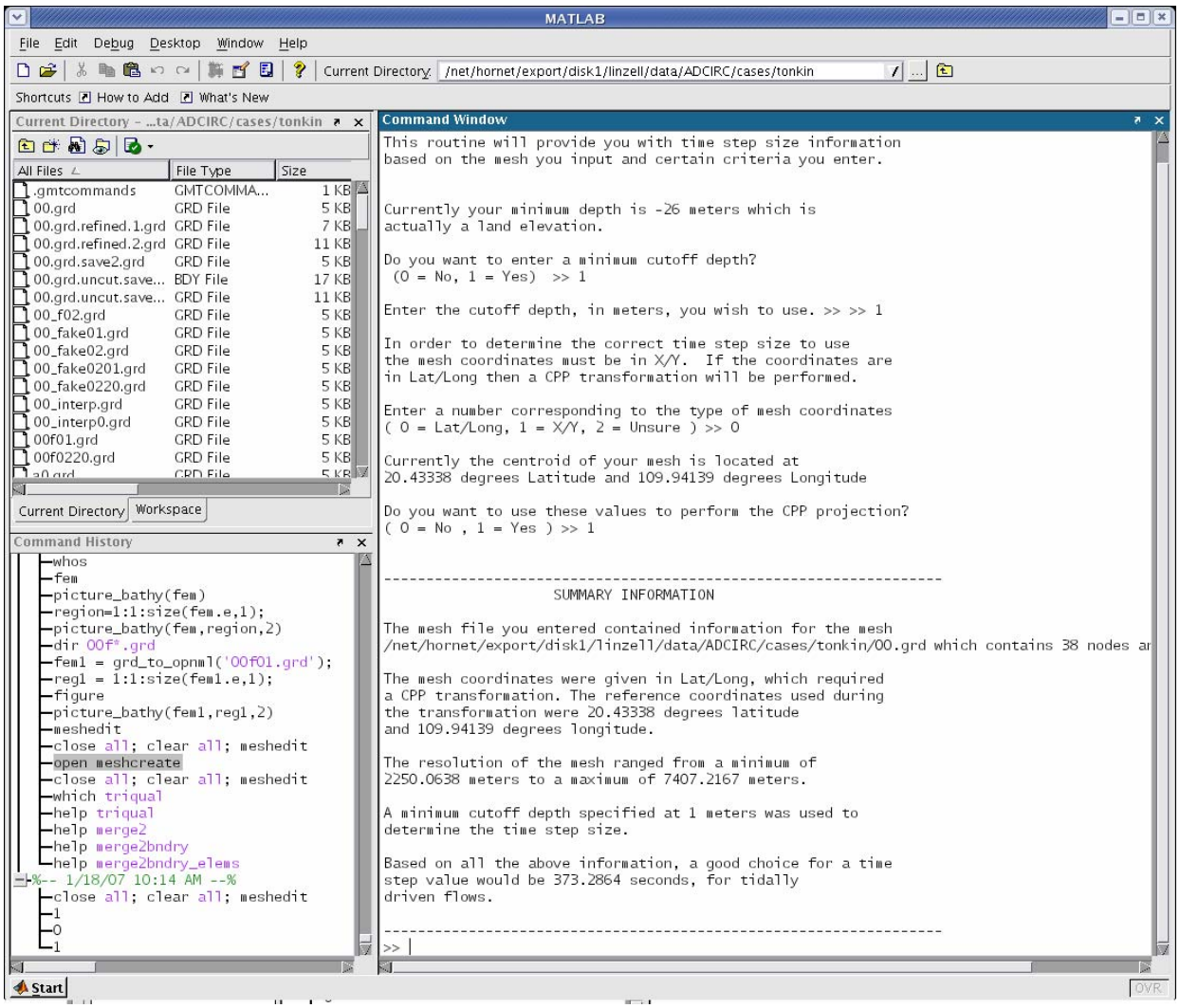


Figure 34. User prompts and calculation summary for the *Delta-T* mesh operation in *meshedit*.

The utility prompts the user whether to enter a minimum cutoff depth:

```
Currently your minimum depth is -1.1496 meters which is
actually a land elevation.

Do you want to enter a minimum cutoff depth?
 (0 = No, 1 = Yes)  >>
```

If **No** is chosen (0 is entered), the minimum depth of the mesh will be used. If a particular depth is preferred (typically 1 m), then enter 1 for **Yes**. The program will then prompt for entry of the cutoff depth value:

```
Enter the cutoff depth, in meters, you wish to use. >> 1.0
```

Next, the coordinate system must be indicated. The calculations are performed in Cartesian (i.e., X/Y) coordinates, so if the mesh is in spherical (i.e., latitude/longitude) coordinates, they must be transformed. The user is prompted to indicate the coordinate system of the mesh:

```
In order to determine the correct time step size to use
the mesh coordinates must be in X/Y.  If the coordinates are
in Lat/Long then a CPP transformation will be performed.

Enter a number corresponding to the type of mesh coordinates
( 0 = Lat/Long, 1 = X/Y, 2 = Unsure ) >> 0
```

If the mesh already is in Cartesian coordinates, a 1 is entered. If the user is unsure of the coordinates, then a 2 is entered. In this case, the first three coordinate pairs are printed to the screen and the user is prompted to enter the coordinate system type. If the mesh is in spherical coordinates (0 is entered as shown), then the user is prompted for a central latitude and longitude which serves as the centroid of the coordinate transformation:

```
Currently the centroid of your mesh is located at
44.03245 degrees Latitude and 9.96126 degrees Longitude

Do you want to use these values to perform the CPP projection?
( 0 = No , 1 = Yes ) >> 1

CPP projection used for coordinate transform.
```

If the user wishes to use a different location for the coordinate transformation, then 0 is entered, and the latitude and longitude values are entered as well:

```
Do you want to use these values to perform the CPP projection?
( 0 = No , 1 = Yes ) >> 0

Enter the reference latitude you wish to use. >> 44.0

Enter the reference longitude you wish to use. >> 9.9

You entered 44 degrees latitude and 9.9 degrees longitude.
These values will be used to perform the CPP transformation.
```

Calculations proceed and a summary is printed, as shown here:

```
-----------------------------------------------------------------
                    SUMMARY INFORMATION

The mesh file you entered contained information for the mesh
/home/username/data/my_test.grd which contains 24741 nodes and 47377
elements.

The mesh coordinates were given in Lat/Long, which required
a CPP transformation. The reference coordinates used during
the transformation were 44.03245 degrees latitude
and 9.96126 degrees longitude.

The resolution of the mesh ranged from a minimum of
25.8253 meters to a maximum of 250.261 meters.
```

```
A minimum cutoff depth specified at 1 meters was used to
determine the time step size.

Based on all the above information, a good choice for a time
step value would be 2.3361 seconds, for tidally
driven flows.

-----------------------------------------------------------------
```

The user should examine the summary information to ensure correctness, and note the suggested time step value. This time step value should be considered a maximum. In practice, the time step entered in an ADCIRC fort.15 file generally would be rounded down from the suggested value. For this example, a time step of 2.0 s might be used.

c. Display Options

As earlier noted, the *meshedit* GUI has several options available for the display of mesh features (e.g., Figure 14). Mesh display options are enabled/disabled by toggled check boxes. The available display options include the following features:

- *Num Nodes* – node numbers are shown adjacent to their respective node
- *Num Elements* – element numbers are displayed in the center of their respective element
- *Show Bathy* – color-filled contours of the mesh bathymetry
- *Show Elements* – mesh elements are delineated; this feature is enabled by default.
- *Show Bad Elements* – bad or suspect quality elements, as determined by triangle quality or angle parameters, are marked
- *Show Nodes* – nodes are shown as small blue circles
- *Show Boundary* – mesh boundary is shown; display of the boundary is recommended, particularly if boundary elements or nodes have been modified, or if islands or other geographic features are to be displayed.

If *Show Bad Elements* is enabled, the user is prompted to choose the criterion by which element quality is determined (see Figure 35). If the user chooses "triangle quality," then the triqual input prompt (Figure 8) is displayed. Otherwise a selection of the "max angle" criterion results in a prompt for maximum angle input (Figure 16). Or similarly, if the user chooses "min angle", the minimum angle prompt (Figure 17) is displayed. Once the user interaction is complete, the *meshedit* software scans the entire mesh for the user-selected criterion. Any elements that fall outside the bounds of the criterion are marked with a red asterisk symbol in the display window as seen in Figure 36 (lower left and right corners of the displayed mesh).

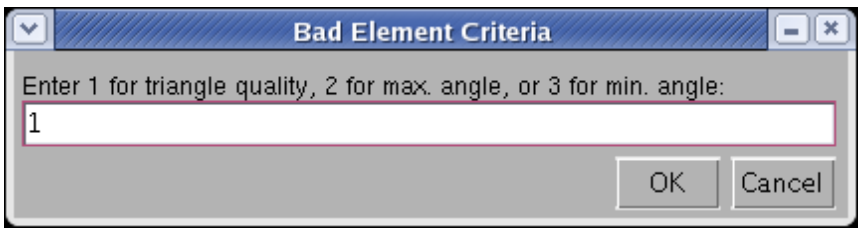


Figure 32. The *Show Bad Element* user prompt for criterion selection; options include the triangle quality, the maximum angle or the minimum angle.

A sample display with most of the features enabled is presented in Figure 36. The “Show Bathy” option is shown by example in Figure 37.

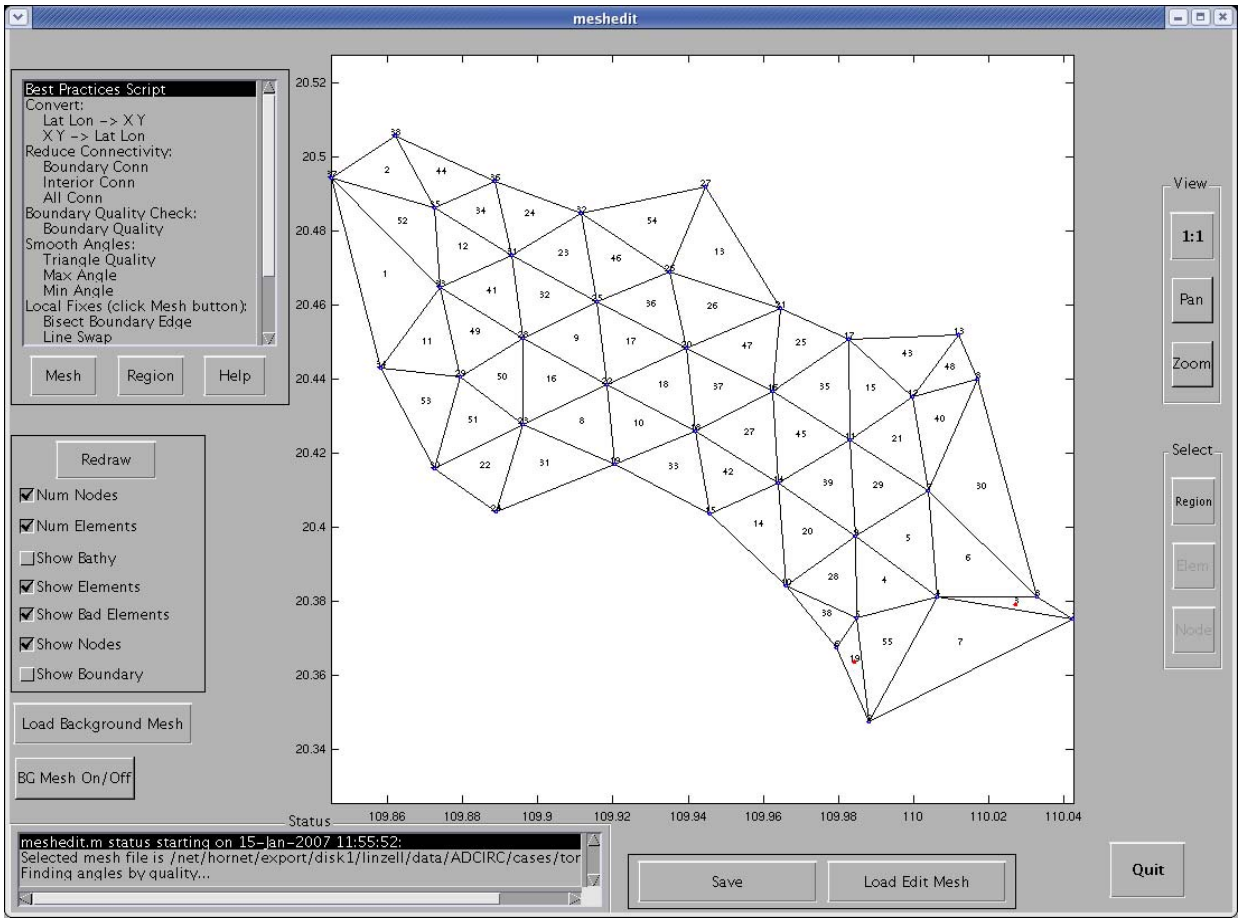


Figure 36. The *meshedit* GUI with display options enabled for node numbers, element numbers, elements, bad elements, and nodes.

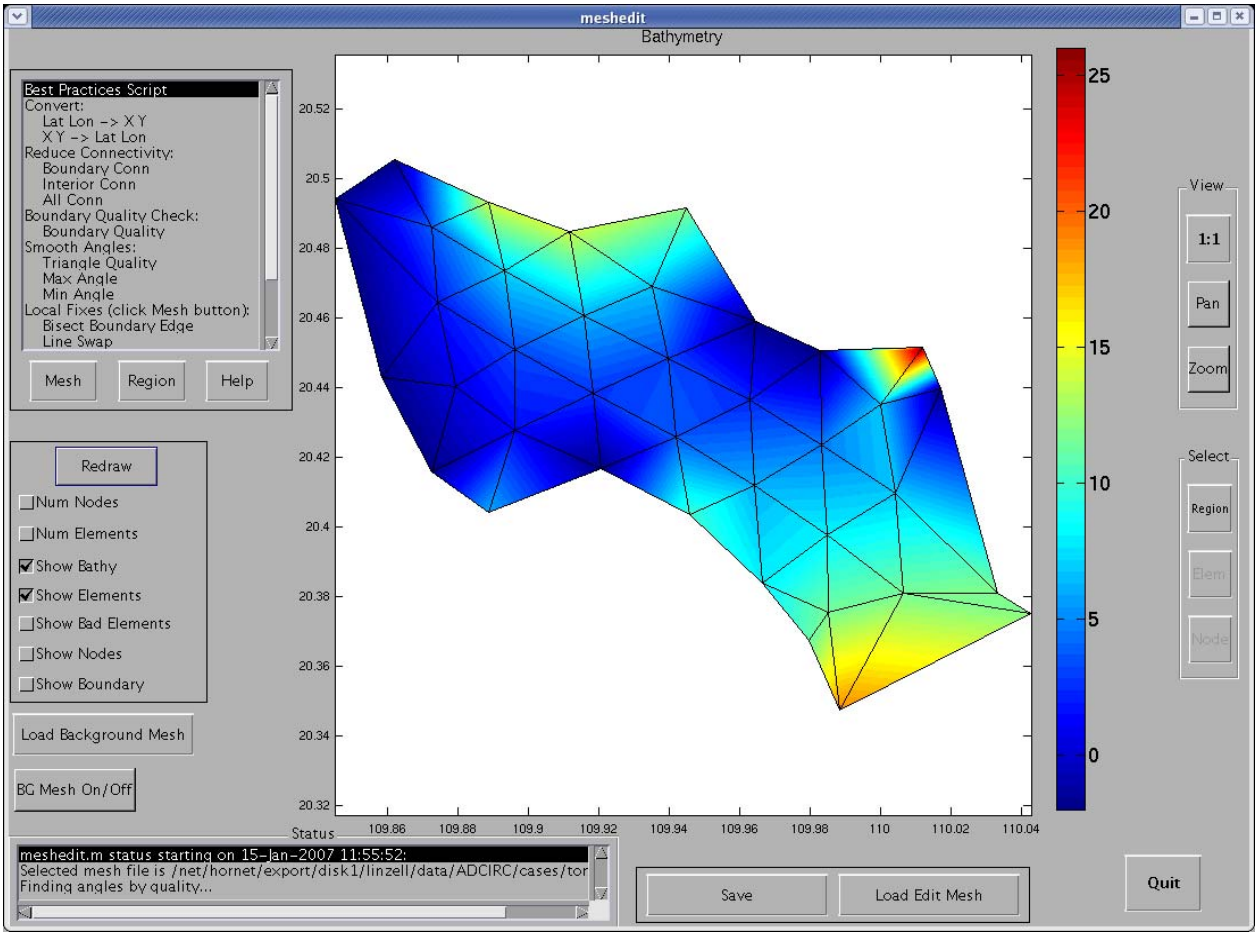


Figure 37. The *meshedit* GUI with the display option enabled for bathymetry contours.

- *Displaying a Background Mesh* – load and display an existing background mesh.

The creation of a background mesh was previously discussed in Section 3b, *Interpolate Bathymetry*. If the user clicks the **Load Background Mesh** button (e.g., Figure 37), a file selection dialog box (e.g., Figure 13) appears, and the user selects a file to load. The software attempts to determine the coordinate system of the background mesh, and if it identifies the coordinate system for the background mesh to be different from that of the edit mesh, the user is prompted for a decision on converting the background mesh (Figure 38).

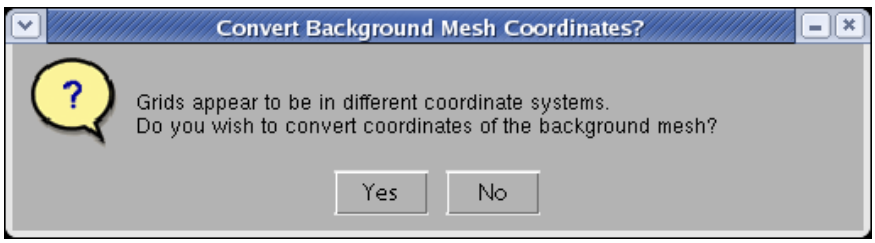


Figure 38. The **Load Background Mesh** user prompt to convert the coordinates of the background mesh.

If the user selects the **Yes** button shown in Figure 38, another prompt is issued to verify the background mesh coordinates. This prompt is shown in Figure 39.

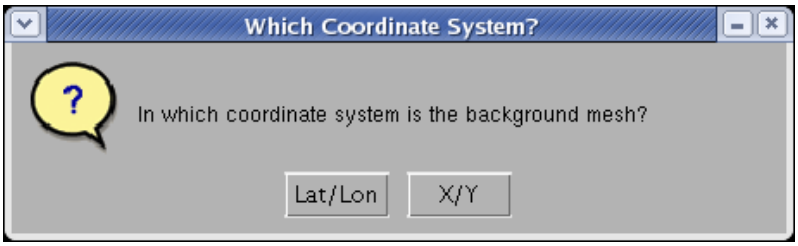


Figure 39. The **Load Background Mesh** user prompt for type of the background mesh coordinate system.

The coordinate choices for the background mesh are Lat/Lon and X/Y. Once the user selects the appropriate type of coordinates, and the coordinate conversion is applied, if necessary. A **No** button selection means no coordinate conversion is done. The **No** option typically only makes sense if the user is certain that the background and edit meshes are represented in the same coordinate system.

Following any necessary coordinate conversion, the background mesh (shown in red) is displayed together with the edit mesh (shown in black) in the plot area of the *meshedit* GUI (Figure 40). The coordinates of the plot area are adjusted to display both meshes in their entirety.

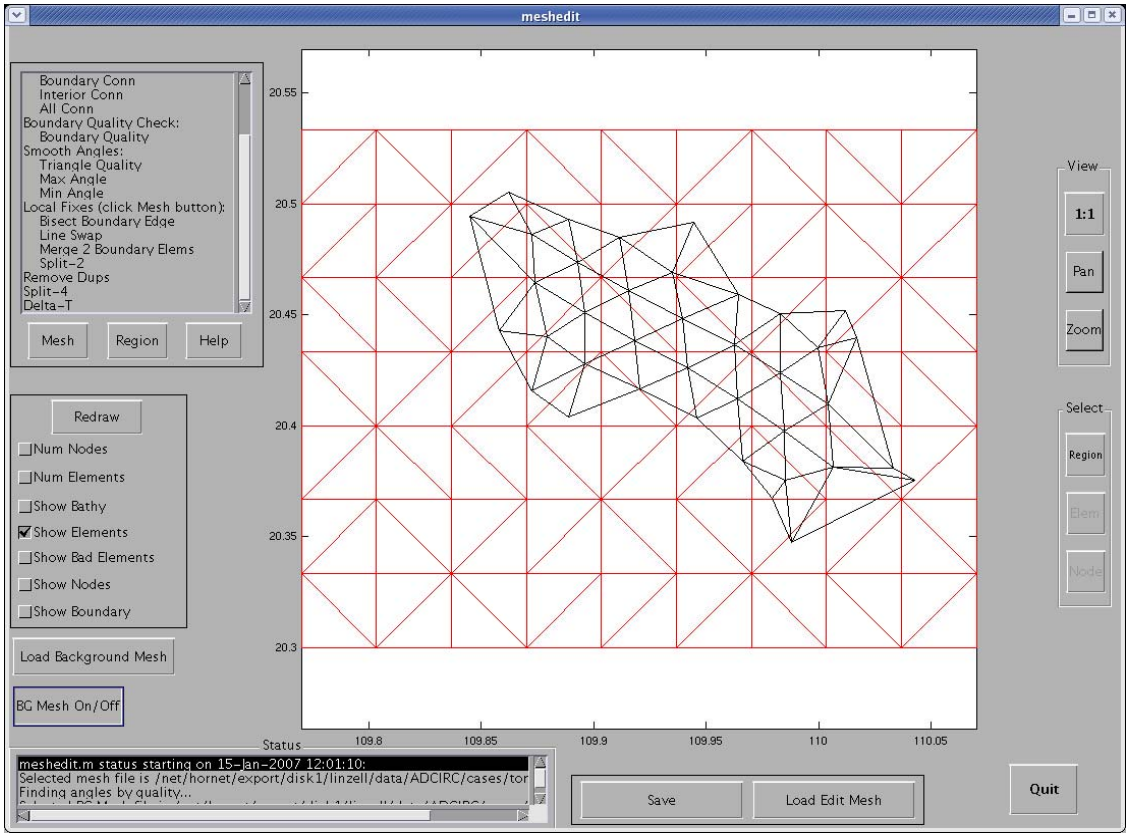


Figure 40. The MeshEdit GUI display of a background mesh (red) and an edit mesh (black).

- *Modifying the Mesh View* – display controls for zoom and pan

Meshedit **View** features (located along the right columnar panel of the GUI) permit the user to zoom into any region of the mesh, or pan the view to examine other portions of the mesh as desired. The user clicks the **Zoom** button (e.g., Figure 40) and a small crosshair cursor appears in the plot area. If the user clicks on a location in the plot area, the view is zoomed in with the center of the zoom region identified by the point location where the user clicked. If the user clicks in the plot area and moves mouse pointer while holding down the left mouse button (commonly referred to as "rubber banding"), the rectangular area traversed by the mouse pointer becomes the new zoomed-in region. Figure 41 shows an example of a zoomed-in region of the edit mesh. Note the plot coordinates are adjusted to display the zoomed-in region.

If the user clicks the **Pan** button, the visible region can be changed by clicking in the plot area, and while holding down the left mouse button, moving the mouse pointer in the direction *opposite* of the desired viewing area. This action has the effect of dragging the edit mesh so that the new region of interest is brought into view. For a large edit mesh or a view that has been zoomed, several successive pans (by repeatedly dragging with the mouse) may be required to bring into view the new region of interest.

To reset the view to the entire mesh, the user clicks the **1:1** button (e.g., Figure 41). This button also can reset the view if toggling of the mesh display features results in an undesirable image.

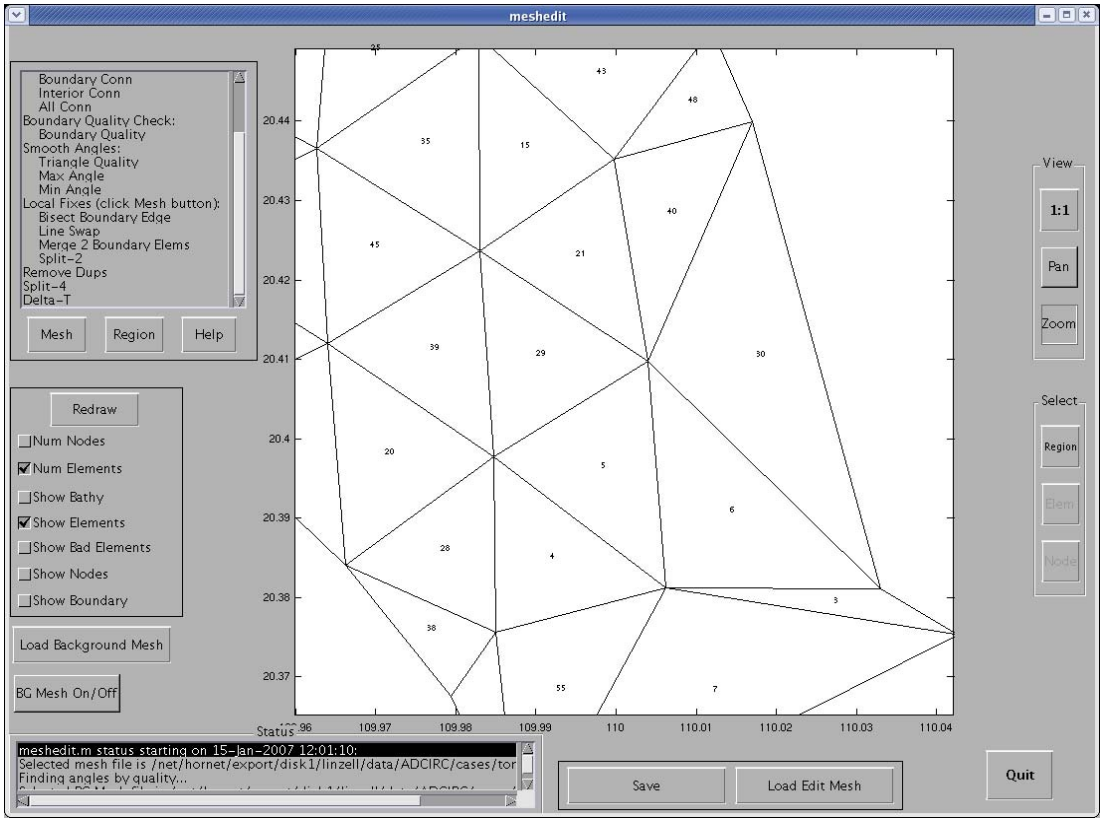


Figure 41. MeshEdit GUI display of a zoomed-in view of mesh after the **Zoom** button was clicked.

4. Limitations

a. Issues

There is a known issue within *meshcreate* and *meshedit* that, because of their in-memory manipulation of grids and the computationally intensive nature of grid manipulation, creates a dependence on system resources. Because of this, large, highly refined grids may cause *meshcreate* or *meshedit* to crash if system resources (memory, CPU utilization) are completely consumed during any portion of the mesh generation or mesh editing process. Windows platforms in particular are problematic.As earlier noted, it is recommended that the user save the created or edited mesh at regular intervals so that modifications are not lost if MATLAB® runs out of memory.

This limitation has revealed itself in *meshcreate* when:

- **The ratio between the 'initial spacing between nodes' and the 'number of refinements' is too small.** There is no set ratio to define since this is a system dependent limitation based on available memory. The result of a violation is that all available memory is utilized since the number of nodes in the in-memory mesh increases exponentially during each refinement. To avoid this occurrence, the user must find a compromise between these two settings. To further refine the generated mesh, it is recommended that the user construct a mesh in smaller pieces or refine the mesh in localized regions, external to *meshcreate* as described in Section 2c. Note that good practice requires re-interpolation of the bathymetry data to the final mesh using the Matlab utility, *bath2grd*, as described in Section 2c.

Current issues in *meshedit* occurs as follows:

- The "disappearance" of the edit mesh in the plot area has been encountered when panning or zooming. Additionally there may be occasions in which mesh features (e.g., nodes, node numbers, element number, etc.) are toggled on and off, or other display modifications have been repeatedly enabled and disabled, and the mesh is no longer displayed. If this occurs, it may be possible to reset the display by disabling all of the mesh features except **Show Elements**, clicking the **Redraw** button, and then clicking the **1:1** button. If this does not correctly display the edit mesh, the user may have to exit the program and restart from the last saved edit mesh.

b. Error Messages

meshcreate:

- *WARNING!! You have no bathymetric data within the coastline region! Stopping Mesh Generation!!*

This error indicates a mismatch between the bathymetry data and boundary information files supplied. Check that the coordinate systems of the boundary information and bathymetry data are the same.

meshedit:

Errors and warnings are issued to the MATLAB command line, the GUI Status window, or as GUI pop-ups.

APPENDIX I: File Formats

a. Boundary File Format for *Meshcreate*

```
filename
total_num_segment       # mainland + islands
num_nodes segment_type  # for each mainland bdy segment
lon   lat               # CCW ordering
num_nodes segment_type  # for each island
lon   lat               # CW ordering
```

Please note: Coastline and open boundary data is ordered counter-clockwise, however island data is appended after the main boundary information and is in clockwise (CW) order.

b. Bathymetry File Format for *Meshcreate*

```
lon   lat   depth       # bathy is +, ordering not required
```

Bathymetry data must be contained within a single file and span the entire region outlined by the boundary file data. The bathymetry file data is not required to be gridded in any particular manner.

c. Grid File Format (ADCIRC) (*fort.14 – boundary type information*)

Input variables listed by line sequence within the file:

```
AGRID
NE, NP
JKI, X(JKI),Y(JKI), DP(JKI) , JKI = 1,NP
JKI, NHY, NM(JKI,1), NM(JKI,2), NM(JKI,3), JKI=1,NE
```

Table Ia. Description of the Variable Names Associated with the ADCIRC fort.14 File (source http://adcirc.org)

Variable	Type	Description
AGRID	Character	Alphanumeric grid identification (<=24 characters).
DP(JKI), JKI=1,NP	Real	Bathymetric value. Bathymetric values are with respect to the Geoid and are positive below the Geoid and negative above the Geoid. Bathymetric values above the Geoid or any depth sufficiently small that nodes will dry, requires that the user enable the wetting/drying feature (NOLIFA=2) in the UNIT 15 input file. Nodes must be input in ascending order.
JKI	Real	Index.
NE	Integer	Number of elements.
NETA	Integer	Total number of elevation specified boundary nodes.
NHY	Integer	Element type. Note that the element type is not an active variable and that only 3 node linear triangles are operational in this version of the code.

Variable	Type	Description
NM(JKI,1), NM(JKI,2), NM(JKI,3); JKI=1, NE	Integer	Element connectivity of node 1, node 2, and node 3 specified with a counterclockwise orientation. Elements must be read in ascending order.
NP	Integer	Number of nodal points.
X(JKI),Y(JKI), JKI=1,NP	Real	X and Y coordinates. If ICS=1 in UNIT 15 then X, Y represent standard Cartesian coordinates specified in length units consistent with other UNIT 15 input (typically meters or feet). If ICS=2 in UNIT 15, then X, Y represent degrees longitude (degrees east of Greenwich is positive and degrees west of Greenwich is negative) and degrees latitude (degrees north of the equator being positive and degrees south of the equator is negative) respectively.

APPENDIX II: Creating Input Data for *Meshcreate*

Since unique boundary and bathymetry data are required to create a new unstructured mesh using MeshGUI, supplemental information is provided on methods available for obtaining these required data files. For convenience, a global data set of bathymetry at 2 minute resolution (NRL-DBDB2) is provided along with Fortran extraction routines to obtain properly formatted bathymetric data over a desired Lon/Lat region.

Two approaches are presented for obtaining boundary information in a format appropriate for mesh generation using MeshGUI. First, a developmental MATLAB utility, *makecoast*, is provided to facilitate the processing of segmented coastline data into the form required by *meshcreate*. A second approach employs a developed Perl conversion routine, *cst2bdy.pl*, which accepts shoreline data created by the SurfaceWater Modeling System (SMS) and converts it to a format appropriate for the *meshcreate* software.

a. Bathymetric Data: Extraction from the NRL-DBDB2 Database

1) Compile the Fortran programs *cutDBDB2.f* and *read_cutDBDB2.f*, found in the DBDB2 data directory to obtain the binary executable files, *cutDBDB2* and *read_cutDBDB2*

2) To extract the bathymetry data, type:

```
> cutDBDB2 LonW LonE LatS LatN
```

where `LonW LonE LatS LatN` define the bounds of your grid domain.

Output from the extraction is the data file: *cutDBDB2.dat*

NOTE: The bathymetry data must cover the entire region enclosed by the boundary data.

3) To reformat the data for *meshcreate* input, type:

> *read_cutDBDB2*

Output is an ASCII data file, *fort.12*, containing three columns that represent location (in X/Y or Lon/Lat) and the depth at that location. The *fort.12* file is in the bathymetry data file format for *meshcreate*. See APPENDIX I for details on the bathymetry data file format.

.

4) More information about the NRL-DBDB2 global bathymetric database can be found in the README file contained in the DBDB2 data directory and on the web at http://www7320.nrlssc.navy.mil/DBDB2_WWW

b. Coastline Generation: Using *makecoast*

1) Obtain raw segmented shoreline data. One source is from NOAA: http://rimmer.ngdc.noaa.gov/mgg/coast/getcoast.html.
   a. Select the region you want

b. Be sure to note the input coordinates for later use!
c. Choose a resolution that will work best, 1:2,000,000 is probably the best for most applications
d. Request the data to be saved in the Matlab format.
e. Download the data file to your local computer.

2) Start MATLAB®

3) Type *makecoast* in the command window

4) Read the downloaded segmented shoreline data file:

```
c = load('file.dat');
```

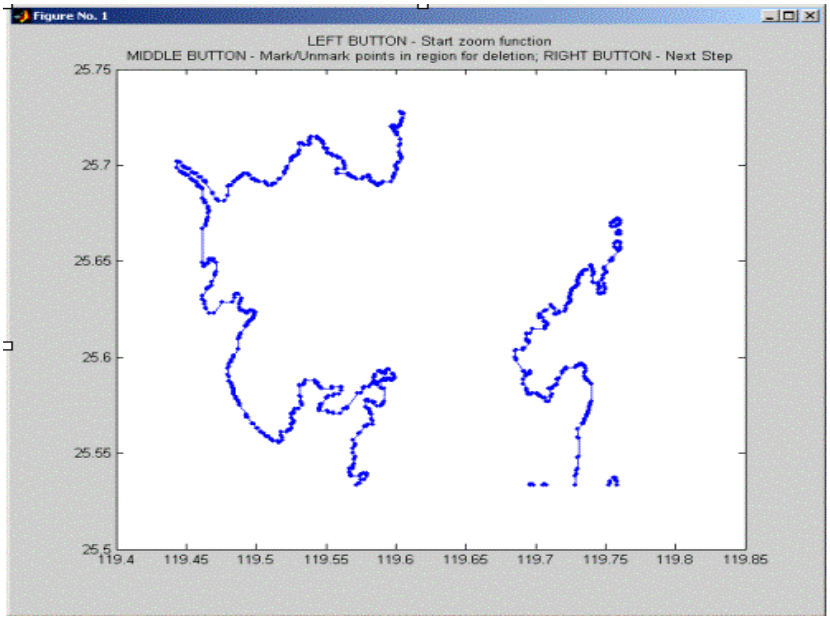
5) Execute the *makecoast* utility:

```
makecoast(c,0.000001);
```

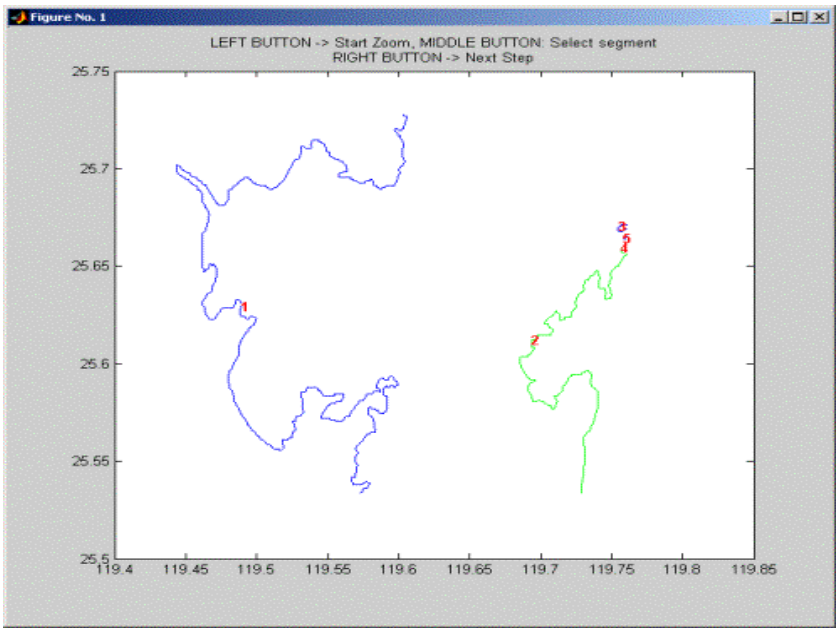
NOTE: the second argument, "0.000001", is a tolerance used to determine the minimum distance between connected points. A very small value for the tolerance is recommended, especially with higher resolution coastline data.

6) Follow the dialogue at the top of each screen:

a. Select nodes to remove:

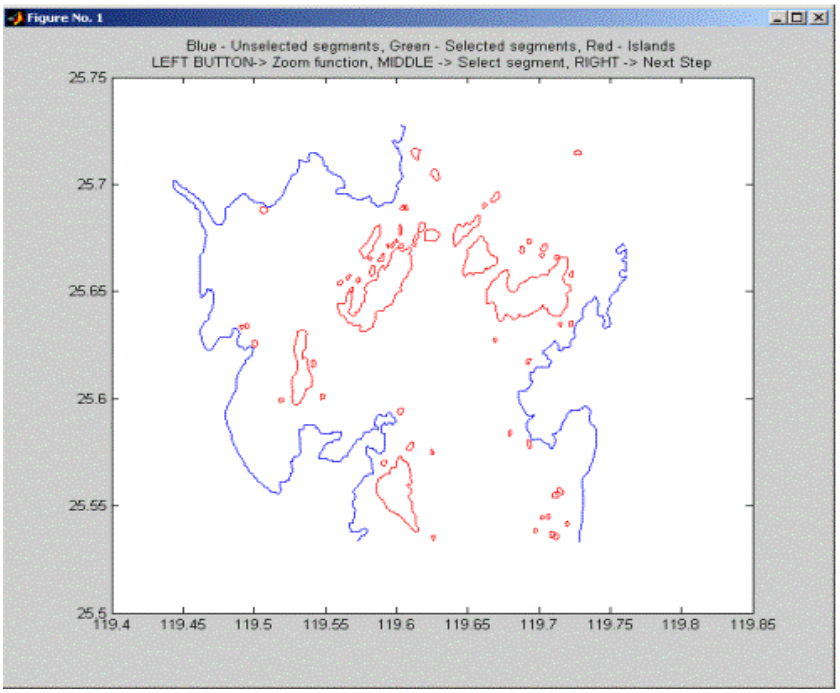


b. Select line segments to connect:

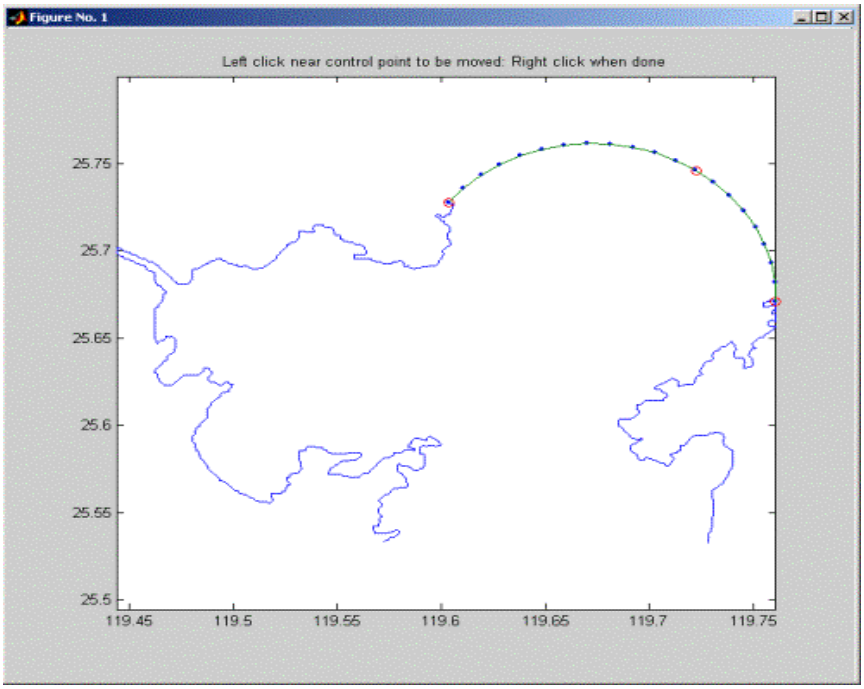


NOTE: The fewer line segments the better. Ideally, you should end up with one or two segments depending upon the number of open water boundaries. This will improve island detection and open boundary creation within *makecoast*.

c. Manage islands by deleting unwanted islands:



d. Define open boundary segments. When using the circular arc, always select points in a counterclockwise order:



Several options are available for specification of the open boundary position (s). Open boundary (ies) can assume the following shapes: arc (default), line, or spline (for more complicated open boundaries).

e. Save the new boundary file.

7) For an explanation of the boundary file format, please see the Appendix I.

8) Troubleshooting

- Errors concerning external boundary segments indicate that too many segments were left to be combined or perhaps a coarser resolution database should be used due to the geometric complexity of the fine-scale data.

- Final adjustments, including the deletion of islands outside of the open boundary, may result in errors. Most problems can be fixed by minimizing the number of line segments, decreasing the tolerance argument and/or using a lower resolution raw shoreline data.

- If the circular arc open boundary points were not entered in a counter-clockwise order, these boundary nodes will have to be reversed in the file.

- *??? Error while evaluating uicontrol Callback.* This error is indicates that the program ran out of memory.

c. Shoreline Conversion from SMS: Using *cst2bdy.pl*

The Perl script, *cst2bdy.pl*, converts the Surface Modeling Software (SMS) shoreline file to an NRL MeshGUI-compatible formated boundary information file. Note that the ordering of shoreline coordinates is preserved, but ordering of the shoreline segments may be modified for compatibility with the software.

Usage:

The conversion tool is run from the command line:

```
> cst2bdy.pl SMS_File > MeshGUI_Boundary_Information_File
```

where: `SMS_File` is the name of the SMS shoreline file, and
`MeshGUI_Boundary_Information_File` is the name of the converted file.

The boundary file created by *cst2bdy.pl* can be read and displayed by *XmGREDIT*, though warnings about the ordering of the coordinates within boundary segments may be issued, but they are of no consequence.

APPENDIX III: Localized Refinement: Using *XmGREDIT*

A third-party software, *XmGREDIT*, is a graphical user interface (GUI) based mesh editing tool that can be used to interactively refine or edit specific areas within a created grid. The resulting grid can then be directly imported back into *meshcreate* or *meshedit* in order to continue with additional refinements, boundary cutting and/or to re-interpolation of the bathymetric data onto the new grid.

XmGREDIT, is installed and executed independently from the MATLAB® codes described in the main body of this document. The code and installation instructions can be found on the World Wide Web (WWW). Examples of *XmGREDIT* usage are described below.

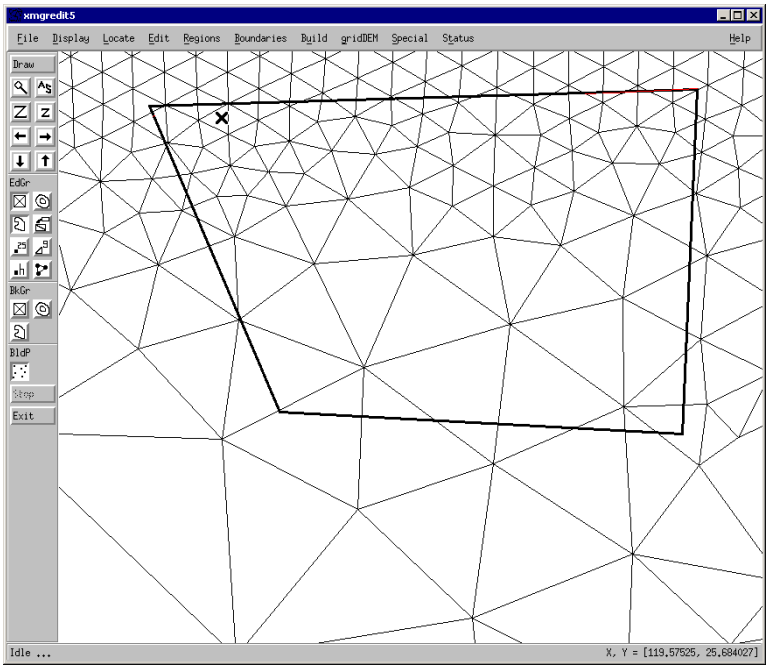
Typically, the depth refinement approach employed by *meshcreate* is deficient in adequately capturing the geometry of deep, narrow channels. As such it is necessary to refine such areas within the mesh using *xmgredit*'s interactive capability for regional refinement:

1) Start *xmgredit* using the final uncut rectangular mesh saved by *meshcreate*:

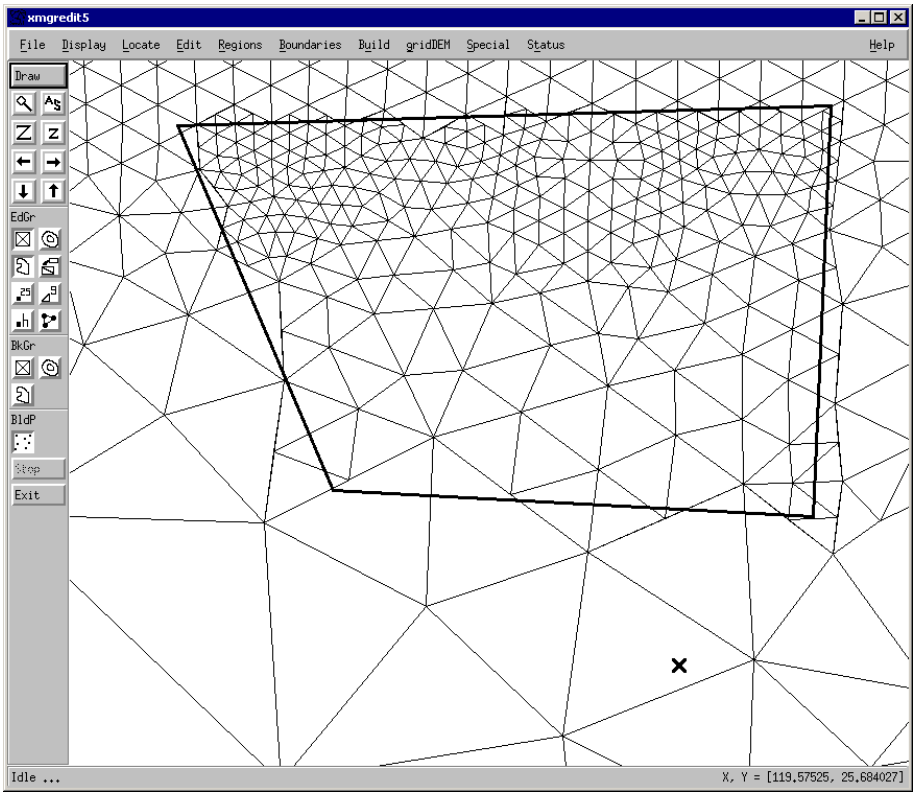
   ```
   xmgredit5 <grid name>.uncut.save1.grd &
   ```

   The file `<grid name>.uncut.save1.grd` is the refined grid within *meshcreate* prior to the removal of nodes and elements outside of the supplied boundary information file.

2) Within *XmGREDIT*, define the region over which to refine
   i. Open grid to be refined in *xmgredit*
   ii. Select "Region >> Create Region" from menu
   iii. Click with the left mouse button to create the points that define the region
   iv. Click with the right [*or middle in some 3 button types*] mouse button to end the process; the defined region should then be enclosed in a thick, black line.



3) Within *XmGREDIT*, refine elements in the selected region
    i. Select "Edit >> Edit Over Grid/Regions" from menu
    ii. Select "Split Elements…" from Edit Region window
    iii. The "Preview" button allows viewing of elements to be refined. Press "Accept" to actually refine the region.



4) Once editing within *xmgredit* is complete, save the *xmgredit*-modified grid as an *Edit Grid*.
5) Startup *meshcreate* in MATLAB®
6) Once in *meshcreate*, select the original boundary and bathymetry data files used in creating the original grid, `<grid name>.uncut.save1.grd`.
7) Press "Refine Existing Mesh" and select the *xmgredit* modified grid as saved in 4) above.
8) Provide a name to which the final refined mesh will be saved.
9) Repeat as necessary to achieve the desired resolution and geometric representations in the final grid.

NOTE: Grids that have been refined AND remain uncut by the boundary data (i.e., retain their rectangular bounding box) are prone to require resources that may be beyond the current *meshcreate* capability. If this approach proves to be too computationally intensive for the available computer resources, consider modifying the final grid (i.e., the non-rectangular, "cut" grid) and re-interpolating the bathymetry using *bath2grd*.

APPENDIX IV: Providing Feedback

All feedback should be sent to **both** of the following NRL POCs via email:

Mr. Robert S. Linzell <linzell@nrlssc.navy.mil> (handles software errors, help questions)
Dr. Cheryl Ann Blain <blain@nrlssc.navy.mil> (handles project level decisions)

Any feedback should include the following information:

1) Provide detailed description about the nature of the feedback:
    a. Error report
        i. Program that caused the error
        ii. Platform details (operating system, version of MATLAB® (if applicable), number of CPU and speed, and available system memory)
        iii. Error message created by the program (verbatim)
        iv. Is the error reproducible? If so, provide details and any specific files (as attachments) that will reproduce the error.
    b. Feature suggestion
        i. Program that the suggestion is for
        ii. Detailed description of the suggestion
        iii. Justification or description of why this feature would be an improvement
    c. Help question
        i. Program in question
        ii. Platform details (operating system, version of MATLAB® (if applicable), number of CPU and speed, and available system memory)
        iii. Detailed explanation of problem or question
    d. Documentation
        i. Details of feedback
        ii. Location in documentation pertaining to feedback
    e. Other
        i. As many details as possible
2) What impact does this issue have on usage of *meshgui*?
    a. Severe
    b. Some
    c. None
    d. Other
3) What is the importance of this feedback?
    a. Urgent
    b. Important, but not urgent
    c. Not important, can wait
    d. Other

Please note:
Feedback will be handled in a timely manner, but NRL reserves the right to determine if the issue raised in the feedback is an immediately actionable item, and if/when this issue will be addressed. If a conflict arises in addressing actionable items, all final decisions will be deferred to the Transition Panel.