# Triangle-Quadrilateral Grid Generation (TQGG) User Manual

-

# A computer program for constructing unstructured grids

Roy Walters
Cascadia Coast Research Ltd.
Victoria, BC Canada
http://cascadiacoast.com

and with help from many others

1 December 2016

# Table of Contents

# Contents

# 1 Installation

## 1.1 Background

TQGG (pronounced like Jacuzzi) is an open source general purpose interactive program that creates 2-dimensional unstructured grids. The long name is Triangle Quadrilateral Grid Generation and it was developed from the program Trigrid and its later incarnation GridGen.

For Linux and OSX, the GUI uses openMotif and is compiled with gcc and gfortran. For Windows, the GUI uses the QUICKWIN routines in Visual Fortran (a combination of Visual Studio and the Intel fortran compiler). When initially developed 30 odd years ago, the program was written in ancient fortran and used a command line interface. With the advent of more modern GUIs, the graphical interface is written in C and has has been separated from the fortran computational code. Hence, this is a mixed-language program.

The source code is in a git repository. The first step is to install git (a revision control program) on the target system. This is freely available software and may be obtained at

http://git-scm.com/downloads

Once git is installed, create a base directory in your user space for the TQGG source; ie, MyPrograms. Change directory to the directory that has just been created. Then type

git clone https://github.com/rrusk/TQGG.git

which will create a directory ./TQGG and install the code there. In the next sections, specific instructions are given for building the program with different operating systems.

## 1.2 Linux

### 1.2.1 System Requirements

Linux systems require the openMotif development environment, gcc, and gfortran. Normally, gcc is available with the code development environment, and gfortran and Motif are available through the software manager for the particular flavor of Linux being used. If netCDF is desired, the netCDF libraries can be installed with the software manager. The fortran interface is also needed and can be downloaded from the netCDF download site. The fortran files should be built with gfortran to be compatible with the other code. Building the interface is simple and well documented. We have used version 4.2.1. The software has been tested on several versions of openSUSE and Linux Mint.

### 1.2.2 Building

On Linux, just type 'make TQGG' in the directory TQGG and the executable will be built and put into the subdirectory 'bin'. Then typing 'bin/TQGG' will execute the program. There are test files in the directory 'demodata'. Typing 'make clean' will remove the object and module object files.

There are 4 flavours of TQGG that can be built by using one of the following names in the make command:

**TQGG** The normal run version.

**TQGGdbg** The debug version with traceback and bounds checking.

**TQGGnc** The run version with netCDF support.

**TQGGncdbg** The debug version with netCDF support and traceback and bounds checking.

Run the program by copying it from subdirectory bin to somewhere on $PATH or run it from bin.

## 1.3 Apple OSX

### 1.3.1 System Requirements

OSX like Linux requires the openMotif development environment, gcc, and gfortran. These are not normally installed on the proprietary OSX machines nor is there a package manager. OpenMotif can be installed from several sources on the web, gfortran is available from the GNU web site, and gcc (GNU or clang version) comes with OSX development environment depending on OSX version. You will need to install Apple's XCode for the compilers. It is interesting getting all this software to dance together, but the result is worth it. The TQGG code was developed on a Macbook with OSX 10.6, XCode 4, and openMotif 2.1.32-22i and with OSX 10.10.5, XCode 7, and openMotif 2.3.4.

### 1.3.2 Building

**OSX 10.6.8 (and earlier?)**

This was the first attempt to port the program from linux to a macbook. After searching around and taking a few missteps, the following procedure was successful.

1. Install XCode from Apple. This provides gcc.
2. Install or update to the latest version of XQuartz (The X11 system used by OSX).
3. Install gfortran from https://gcc.gnu.org/wiki/GFortranBinariesMacOS.
4. Download the openMotif 2.1.32 libraries and include files from http://www.ist-inc.com/motif/download/index.html

5. Build TQGG by typing 'make -f Makefile-OSX TQGG', where TQGG can have one of the forms described earlier.

Run the program by copying it from subdirectory bin to somewhere on $PATH or run it from bin.

**OSX 10.10.5**

Somewhere between OSX 10.6.8 and 10.10.5, the available download for openMotif 2.1.32-22i no longer functions properly and openMotif must be re-built. However, others have made this a simple task. In all, the necessary software is provided by following these steps:

1. Install XCode and Command Line Tools from Apple. This provides gcc (clang) and the tools for rebuilding openMotif.

2. Install or update to the latest version of XQuartz (The X11 system used by OSX).

3. Install gfortran from https://gcc.gnu.org/wiki/GFortranBinariesMacOS.

4. Install Homebrew for package management from http://brew.sh/. Also see https://github.com/Homebrew/homebrew.

5. Build the openMotif libraries and include files using a one-line command with Homebrew.
   See https://gist.github.com/steakknife/60a39a32ae84e12238a2

6. Create a symbolic link so the compiler and linker can find the openMotif include files and libraries.
   sudo ln -s /usr/local/Cellar/openmotif/2.3.4 /usr/local/OpenMotif

7. Build TQGG by typing 'make -f Makefile-OSX TQGG', where TQGG can have one of the forms described earlier.

Run the program by copying it from subdirectory bin to somewhere on $PATH or run it from bin.

**OSX 10.11**

With this version, the C compiler is now clang which intercepts the gcc call. Unfortunately clang is sometimes incompatible with everything so the gcc compiler supplied with gfortran must be used. If this is necessary, then the last step above is replaced with

1. Build TQGG by typing 'make -f Makefile-OSX2 TQGG', where TQGG can have one of the forms described earlier.

Run the program by copying it from subdirectory bin to somewhere on $PATH or run it from bin.

## 1.4 MS Windows

### 1.4.1 System Requirements

The Windows version requires Visual Studio and a fortran compiler that includes the QUICKWIN libraries. These libraries are fortran callable subroutines that access a simplified version of the WIN API. The program was developed using Win XP SP2, Visual Studio 2005, and Intel fortran compiler 2011.8. Visual Studio 2008 has also been used with 64-bit XP where the project file is automatically updated to a new format.

### 1.4.2 Building

The project file TQGG.vfproj can be found in './TQGGWS/VS2005'. Open this file in Visual Studio and there will be options to build the 32-bit debug and release versions, and the 64-bit debug and release versions.

Unlike the Linux and Mac versions, all the dialog boxes in Windows are modal; ie, the dialog must be closed before any other operations can commence. The only solution to this is to program the GUI directly with the WIN API and not use QUICKWIN.

There have been issues in running these executables on later versions of Windows (7 and possibly 8). Apparently, there is some security issue with the later versions such that the dialog boxes will not accept any input so the program hangs. This may not happen with later versions of VS and QUICKWIN that are built for the later versions of Windows.

# 2 TQGG Program Description

## 2.1 Introduction

This program uses interactive graphics to read geometric and grid data, to create a background grid from the geometric data, to create a model grid, and to allow examination and modification of an existing grid. At the end of an editing session, this program can output a triangle list in addition to the modified grid. The program permits the user to display various properties of the grid, such as coordinates of individual vertices. It also provides means of displaying various properties of vertices and triangles that normally cannot be judged by eye. For instance, colour markers can be placed at all vertices where water depth exceeds a specified value.

Changes can also be made to the grid. For instance, vertices and connections between vertices can be added or deleted; vertices can be moved or merged with one another; and triangle shape can be adjusted. The user directs changes in purely graphical terms, by suitable positioning of a cursor on the displayed grid; the program keeps account of all corresponding changes in vertex coordinates and interconnections between vertices. Any proposed changes are displayed immediately, for confirmation or cancellation.

## 2.2 Use of TQGG

This program is invoked from the command line by typing the name of the executable file (typically 'TQGG'). A frame for the grid editing area is drawn in black on the screen and a menu along the top of the panel allows interaction with the program. At this point, the user chooses a menu item that leads to node or grid input, whichever is desired. The initial display shows the entire grid being edited. If the grid is larger than 1000 nodes, only the outline is shown initially. The entire grid can be displayed by selecting **{View}Redraw** from the View menu.

Editor options are presented to the user in menus that appear across the top of the screen, and additional prompts are displayed when necessary. Selections from the menus are made by means of the mouse, but some editing operations require keyboard input also.

The various editor options are discussed below, menu-by-menu. One important editor facility which should be noted is the save option; it is recommended that the current version of the grid should be saved at regular intervals during long editing sessions, in case of power failures or computer gremlins.

# 3 Grid generation procedures

Generating an unstructured grid relies on a series of subtasks that depend primarily on the study objectives and the data or initial grid available for input. If the study is a one-off, then usually the study area is treated in some detail and the grid becomes coarser out to where the boundary conditions are applied. If

the grid is a basic resource for several current and future studies, then it may be better to create a larger scale background grid with good quality everywhere, and refine the grid in specific areas as necessary.

The initial data for the grid can come from a wide range of sources. Generally, it is easier to start with an existing grid and make modifications than to start from scratch with sampling. Two types of data are required: data that approximate the external and internal boundaries, and data that provide the topography. Here we use a right-hand coordinate system with x directed eastward, y directed northward, and z directed upward, although the (x,y) plane can be rotated if appropriate. The dimensional units are either degrees for spherical polar coordinates or meters otherwise. Arbitrary units can also be used if care is taken to not invoke inappropriate transformations such as UTM or Polar.

The program has sufficient flexibility that tasks can usually be performed using several methods. In the following subsections, we present methods to accomplish some of the major grid generation tasks. The user may find other methods that are more suitable for their problem and that is all in keeping with the design of the program. Other more specific operations can be found while perusing the menu items.

## 3.1 Sampling data

Boundary data is obtained from an existing grid by reading the grid using **[File]/OpenGrid** and then transforming the grid file to a node file using **[GridEdit]/Grid2Nodes**. Save the node file and then read it back using **[File]/OpenNode**. At this point, the data is represented as a series of boundaries starting at the outer boundary, followed by island boundaries followed by line constraints, and finally all the interior nodes. Delete the interior points by defining a polygon with **[Polygon]/Whole** and then **[NodeInPoly]/DeleteInt**. The result is a data set with external boundary, islands (if any), and interior line constraints (if any). Save this file for future use and editing with **[File]/SaveAs**. Generally, it is safer to save a file that has been modified then read it again because this operation cleans up the indexing associated with deleted nodes. At this point, the boundary resolution can be changed in individual sections using **[NodeEdit]/ReSample** or within a polygon using **[NodeInPoly]/ReSample**. The boundaries can also be edited using items in the menu **[NodeEdit]** or **[NodeInPoly]**. Island boundaries and line constraints from other files can be added through the menu item **[File]/AddNode**.

Boundary data can also be obtained from an existing set of coordinates for point data through the menu item **[File]/Sample**. For instance, the input data can be a text file with lines containing (x,y) or (x,y,z) coordinates generated from vector shorelines or other sources. The program will prompt for a minimum distance between data points and then filter the data to reduce the size of high resolution data sets. The minimum distance should be some fraction of the desired final resolution ($\approx 1/4$ or so) in order to maintain accuracy when sampling. Save each input file as a node file using **[File]/SaveAs**. Each output file contains one boundary arranged in order from one end to the other and

the boundaries from each file can be added together using **[File]/AddNode** and adding each boundary in sequence. These boundaries can then be joined, reversed, and manipulated using items in **[NodeEdit]**. As an alternative, the files may include multiple segments with lines containing (x,y,z,s), where s is a segment identifier. All points with the same segment identification should be arranged in order from one end to the other and the points will be automatically arranged into a separate boundary. Multiple segment data can be manipulated in the same manner as the single segment data.

The end product after editing is a node file containing boundaries at the specified resolution. This file is used directly by the grid generation procedures. AN IMPORTANT NOTE: the node spacing along boundaries should vary smoothly otherwise the grid generation procedure that starts on the boundary will generate highly irregular elements corresponding to the degree of irregularity of the boundary data. The resample options provide an easy method to smooth the distance between boundary points. Choosing menu item **[Info]/BoundaryCheck** will run a procedure that will indicate whether there are errors in any of the boundaries. If the data pass this test, the next step is grid generation.

## 3.2 Generating a grid

An important prerequisite to generating a grid is to verify that distances measured along the x and y axis are the same; that is, the aspect ratio (x distance/y distance)=1. For grid units of meters (igridtype=1 or 2), this is satisfied. For user defined units (igridtype=3) this may not be the case so that the x or y dimensions must be scaled so that aspect=1. Otherwise, the elements that are generated will be distorted and stretched in one axis direction, leading to reduced accuracy in a numerical model. This issue also applies to spherical polar coordinates (igridtype=0) since horizontal distances in degrees decrease as the poles are approached. For the latter, this problem is resolved by transforming to a local coordinate system using **[View]/PolarTransform** so that the aspect=1. However, after grid generation and before saving the file, the grid must be transformed back to the global polar coordinates by invoking **[View]/PolarTransform** again.

The main grid generation algorithm is a frontal marching method that starts at the boundaries and attempts to create equilateral triangles as it marches into the interior. The size of the triangles can be controlled in 2 ways. In the first case if no reference grid is chosen, the size will vary linearly between boundaries. This gives a relatively uniform element size that grades from the inner boundaries to coarser open boundaries. In the second case if a reference grid is chosen, the depth in the reference file is used to determine element size; ie, deep areas have larger elements than shallow areas. Limits are placed on the rate the element size can grow or decrease in order to maintain a smooth variation in sizes.

The overall procedure then is to read in the node boundary data using **[File]/OpenNode**. Then use **[GridGen]/OneFront** in order to examine the results from the generation of the first front. If these results are acceptable, then read the node file again and select **[GridGen]/AllFronts**. The program

will then generate interior elements throughout the area within the external and island boundaries.

[Under construction]

### 3.3 Local refinement

[Under construction]

### 3.4 Grid quality

The overall objective is to generate an unstructured grid that is free of errors, grades smoothly from one area to the next, and contains roughly equilateral triangles and/or square quadrilaterals. Errors can be eliminated completely but the grading and element shape can only be optimized due to geometric constraints.

[Under construction]

## 4 Menu description

The top menu has entries that lead to similar groups of operations. For instance, **File** contains the operations to read and write data into files and **View** contains the operations to change the display (zoom, pan, etc.). Selecting a top menu item brings up a submenu with more specific operations. An attempt has been made to make the menus and submenus both logical and intuitive.

### 4.1 Contents of top menu

When the program begins, the following top-level menu appears across the top of the screen:

| **File**\|**View**\|**Info**\|**GridGen**\|**NodeEdit**\|**GridEdit**\|**Polygons**\|**NodeInPoly**\|**GridInPoly**\|**Help** |
|---|

The entries in this menu indicate further sub-menus, which can be selected by placing the cursor on the appropriate word in the menu and then clicking the mouse, i.e. pressing any button on the mouse.

In the following discussion, each menu item is prefixed or followed by a reminder in curly brackets of which menu the option appears in, e.g. option EditNode in the TOP menu will be referred to as "{TOP}**EditNode**", "**EditNode** in {TOP}" or as **{EditNode}**.

In all editing operations, which involve moving nodes, such as Move or Reshape, depths at new locations of nodes affected by the changes are evaluated automatically by linear interpolation among existing nearby depths. In other cases, the user is offered the choice of setting depth by linear interpolation or by entering a value via the keyboard.

## 4.2 Contents of menu: File

When this option is picked, the following menu options are displayed:

Table 1: File menu items.

| File : |
|---|
| OpenGrid |
| AddGrid |
| OpenNode |
| AddNode |
| Sample |
| XSection |
| IntrimSave |
| SaveAs |
| Quit |

These menu items are described next.

### 4.2.1 Menu item OpenGrid

This option allows for reading a new grid file in NGH or GRD format (see Section 5). Any existing data is replaced.

### 4.2.2 Menu item AddGrid

This option allows for reading a new grid file in NGH format (see Section 5). This grid is merged with any existing data so that this action is used to join grids together. Along the edge of the grids that are merged, the boundary node codes are changed to 90, which make this a line that is fixed in space. After merging files, use the **{Info}NodeCheck** to examine the codes at the ends of the lines where the grids are joined. In many cases, the codes at these points need to be changed to 1, 5, or 6, depending on the type of boundary (land, open, or junction of land and open).

When joining grids in this way, it is helpful to begin the editing session with the two sub-grids displayed in different colours. This is be done by default using a SECONDARY COLOUR INDEX.

The two sub-grids may have to be stitched together as required. Some nodes may have to be deleted if the two sub-grids overlap; conversely, extra nodes may have to be added if there is a substantial gap. Connections between nodes of the two sub-grids may be established using **{GridEdit}AddLine** or the node-merging capability available through **{GridEdit}Merge**. To use this later,

which is very convenient where there are pairs of adjacent nodes from the two sub-grids, it is first necessary to invoke **{GridEdit }GridMerge**.

#### 4.2.3 Menu item OpenNode

This option allows for reading a new node file in NODE format (see Section 5). Any existing data is replaced. After picking this option, the user is prompted for the file name of a NODE format file.

#### 4.2.4 Menu item AddNode

This option permits reading a NODE format file and then adding data from other NODE format files. This data can be island (inner) boundaries, line constraints which are interior nodes that are in fixed positions and linked together, and arbitrary interior nodes. The usual procedure is to read a file with the outer boundary, then select menu item **AddNode** and follow the prompts to add selected data.

#### 4.2.5 Menu item Sample

This menu item is used to sample *.xyz files into TQGG. The workspace will be cleared prior to sampling the files. The user will be prompted to input a gridtype and minimum sample spacing. The sample spacing is defined in meters and will subsample the boundaries by only including new nodes that are farther away from the previous one than the minimum spacing.

#### 4.2.6 Menu item XSection

This option is used for creating grids from cross section data. A file containing cross section data is read and a grid is created from this data (sample data: demodata/sriver.xsc). The user is prompted for the number of nodes to create across the section, the number of nodes to create between each cross section, the type of grid (nodes, triangles, quadrilaterals), and the grid units. All these nodes are interpolated using a cubic spline algorithm. See Section 5 for a description of files in XSEC format.

#### 4.2.7 Menu item InterimSave

Invoking this option leads to output of the current version of the grid in '.NGH' format or node file in a '.NOD' format. The resulting file is less compact than the NGH file and NODE file obtained with the usual EXIT procedure (see below). But it is a useful facility and should be used at regular intervals during a long editing session to avoid losing one's work in the event of power failure or other interruption. Alternate interim saves are written to files named interim1.*** and interim2.***; the name of the last interim save file output can be checked via the **{Info}Files** option.

#### 4.2.8 Menu item SaveAs

This is the option normally used after the completion of a grid or modification of a grid and node file. It brings up a request for the name of the file in which the final version is to be saved.
WINDOWS BEHAVIOUR: If the file does not exist then a prompt appears asking the user whether or not to create the file. If no file output is selected or 'CANCEL' is selected, then the program exits with open error message.

#### 4.2.9 Menu item Quit

This allows exit without output of any files. The user is prompted to answer whether the file is to be saved and whether an exit is really desired. This helps prevent accidental termination of the program.

### 4.3 Contents of menu: View

This option provides control over windowing. When the View menu is chosen from **{TOP}**, the options appear as shown in Table 2.

Table 2: View menu items.

| View: |
|---|
| Redraw |
| Outline |
| FullSize |
| Zoom |
| ZoomOut |
| Pan |
| LastView |
| Scale |
| Shift |
| Rotate |
| PolarTransform |
| UTMTransform |

The functions in this menu provide control over windowing (zoom) and refreshing the display. Up to 50 levels of windowing are allowed.

#### 4.3.1 Menu item Redraw

This function forces a refresh of the current display. It may be used for instance when grid lines have become partially erased during removal of markers. It is also used in some instances to get a clean display after turning off options such as display of the original digitized boundaries.

#### 4.3.2 Menu item Outline

This option sets a switch that allows only the boundary to be drawn. This speeds the redraw considerably and is useful in the manipulation of large grids.

#### 4.3.3 Menu item FullSize

After a set of windowing, this option will bring the display back to the full size displaying the whole gird.

#### 4.3.4 Menu item Zoom

Selection of this option with the mouse permits windowing in (zoom in) on any square sub-area of the current window. The area are specified by assigning to opposite corners of the screen. To specify the corners, click and hold the mouse button, move the mouse to a new location and release. The corners are defined by the location for click and location for release. The new window is automatically squared off and refreshed.

#### 4.3.5 Menu item ZoomOut

When ZoomOut is selected with the mouse, the program will zoom out to the lowest zoom level and refresh the window.

#### 4.3.6 Menu item Pan

This option permits the user to move the current window in discrete steps in any direction over the grid, that is, to display, at the same level of magnification, a portion of the grid adjacent to that currently being displayed. This can be used, for instance, to tour a boundary at a high level of magnification. After choosing Pan, press and hold the mouse button at any location. To specify pan direction and distance, move the mouse in desired direction and distance, and release the mouse button. The window selection will move and refresh accordingly.

#### 4.3.7 Menu item LastView

This section returns the viewing window to its previous state (before a zoom, pan, FullSize, etc. operation).

#### 4.3.8 Menu item Scale

This menu selection allows the x, y, or z dimensions to be linearly scaled. The user will be prompted to input scale factor for each dimension.

#### 4.3.9 Menu item Shift

This allows shifting of the x and y coordinates of all the grid nodes. The user will be prompted to input the amount to add to each dimension.

#### 4.3.10 Menu item Rotate

This selection allows the grid to be rotated by a given angle. The user will be prompted to input the angle of rotation. The grid will be rotated clockwise. After rotating, it might be necessary to adjust zoom or pan the window.

#### 4.3.11 Menu item PolarTransform

This selection converts the present x and y coordinate using a spherical polar transform. Selecting this menu item once performs the forward conversion (from degree to spherical polar coordinates). Selecting this item a second time performs the inverse operation.

#### 4.3.12 Menu item UTMTransform

This selection converts the present x and y coordinate using a UTM Transform. If the current grid type is Lat/Lon it will convert to UTM. If the current grid type is UTM, it will convert to Lat/Lon. The user will be prompted to input a UTM zone for the conversion.

### 4.4 Contents of menu: Info

When the Info menu is chosen from **{TOP}**, the options appear as shown in Table 3.

These permit placing of coloured markers on the screen to mark locations or to display properties of vertices and triangles. Solid colouring of triangles is available as an alternative to colour markers.

#### 4.4.1 Menu item NodeInfo

Selection of this menu item prompt the user to select a node. In the Motif version of TQGG this is done using the mouse to click on the node. The following information is displayed in the terminal window.

- Index number
- Node Code
- Coordinates (X, Y, Z)

Table 3: Info menu items.

| Info: |
|---|
| NodeInfo |
| ElementInfo |
| NodeCheck |
| ElementCheck |
| BoundaryCheck |
| EraseChecks |
| PMarkers |
| EraseLast |
| EraseAll |
| SetRange |
| TooClose |
| Files |
| Limits |

A marker is placed at each vertex examined. If a number of points are examined, it may be desirable to erase existing markers in order to be able to spot new markers. **{View}Redraw** erases all markers.

After choosing a node, the user may change its depth or computational code by changing the corresponding current values shown in the dialog box and picking update.

#### 4.4.2 Menu item ElementInfo

Selection of this option prompts the user to select an element. This is done using the mouse to click on the element. The following information on the selected element appears in the dialog box:

- Index number
- Coordinates
- Element code (type)
- List of nodes in the element

A marker is placed in each element examined. If a number of them are examined, it may be desirable to erase existing markers in order to be able to spot new markers. **{View}Redraw** erases all markers.

The element code can be changed by changing the appropriate value in the dialog box and selecting update. To find an element by index, change the index number to the desired value and select update.

### 4.4.3 Menu item NodeCheck

This enables labeling of grid vertices with coloured markers according to certain built-in criteria or an external list, thus providing an efficient visual means of simultaneously checking specific properties at all vertices visible in the window. When this option is selected, appropriate information appears in a dialogue box with available criteria as shown below.

Table 4: Vertex criterion

| Option | Description | Default |
|---|---|---|
| C0 | Computational code equals 0 | OFF |
| C1 | Computational code equals 1 | OFF |
| C2 | Computational code equals 2 | OFF |
| C3 | Computational code equals 3 | OFF |
| C4 | Computational code equals 4 | OFF |
| C5 | Computational code equals 5 | OFF |
| C6 | Computational code equals 6 | OFF |
| C7 | Computational code equals 7 | OFF |
| C8 | Computational code equals 8 | OFF |
| C9 | Computational code equals 9 | OFF |
| NC0 | Computational code not equal to 0 | OFF |
| C=? | Computational code to be set by user | OFF |
| | | |
| DLT | Depth less than d1 | OFF |
| DGT | Depth greater than d2 | OFF |
| DBTW | Depth greater than or equal to d3 and less than or equal to d4 | OFF |
| | | |
| NBGT | No of neighbours greater than n1 | OFF |
| NBLT | No of neighbours less than n2 | OFF |
| NBE | No of neighbours equals n3 | OFF |
| EXT | Mark according to external list | OFF |

The final option EXT instructs the program to read in a prepared external file (in EXTVER format), each line of which contains the number of a vertex where a marker is to be placed, followed by an integer specifying the colour of the marker. [This option not yet implemented in Motif version of TQGG].]

The name of the external criterion file is not requested until the EXT option is invoked; consequently, it is possible for the user to colour vertices according to different criteria in succession by preparing as many external files as required. The name of the external file currently assigned is displayed in the information panel. A different external file can be assigned by clicking the name of the existing file in the information panel.

Finally, when all details of the criterion or criteria to be displayed have been decided, the user should press the "Run Check" button in order to view the appropriate coloured markers.

#### 4.4.4 Menu item ElementCheck

To permit convenient monitoring of certain triangle properties, default or user defined colour tables are used to place coloured markers and solid colours in triangles. When the colouring option is invoked, necessary list of all triangles and their calculated triangular properties are updated. This enables the user to check effects on triangle properties of any editing operations immediately.

When the ElementCheck option is selected, a dialogue box appears with options for selecting the colouring mode and the type of test as shown below.

| Colour Triangles By Criteria |
|---|
| Select Colouring Mode: |
| Full Color |
| Color Marker |
| Select Criteria |
| EQL |
| DEP |
| A2D |
| CCW |
| G90 |
| COD |

The colouring mode can be selected between "Full Colour" and "Colour Marker" with the appropriate radio buttons. "Colour Marker" indicates that coloured markers will be placed in the triangles, while "Full Colour" indicates solid colouring of triangles. Markers are preferable if editing operations are to be carried out on the grid.

Picking the radio buttons beside EQL, DEP, A2D, CCW, G90, or COD determines which of the following internally evaluated triangle properties is to be displayed:

- EQL - measure of equilateral shape, defined as the ratio of the sum of the squares of the sides of the triangle-to-triangle area, normalized in such a way that an equilateral triangle has this ratio equal to unity. This ratio or shape factor increases as triangle shape departs from equilateral. For instance, a right-angled isosceles triangle has a shape factor of 1.154
- DEP - mean depth (average of depths at vertices)
- A2D - area/mean depth
- CCW - clockwise test (+1 if vertices ordered counter clockwise in triangle list, -1 if clockwise). The default colour scale for counter clockwise is black and red for clockwise.
- G90 - flags triangles with angles greater than 90 degrees. The default colour scale is black if less than 90-degree angles, red otherwise.
- COD - element code. The default colour scale follows the colour indices.

In addition, an external file (in EXTCRI format) containing a list of triangles and corresponding values of any quantity defined by the user can be read in, using option EXT.

After the needed check and options are selected, the check can be run by pressing RUN CHECK. The markers or coloured triangles will be displayed. To erase the checks, use **{Info}EraseChecks**

#### 4.4.5 Menu item BoundaryCheck

Selection of this option will perform boundary checks on the boundary. The check is intended to be run prior to doing a triangulation to make sure that the grid is defined correctly.The following checks will be performed:

**Boundary Orientation** All boundary orientations are checked. Clockwise boundaries are then displayed in *red*, while counter-clockwise boundaries are display in *green*

**Outer Boundary Location** The outer boundary is found and if this is not the first boundary in the data-arrays, the user will be prompted to move it.

**Outer Boundary Orientation** The outer boundarys orientation should be CCW. If it is CW, the user is prompted to reverse it.

**Island Boundary Orientation** All other boundaries than the outer boundary should be defined CW. If any are CCW, the user will be prompted to reverse them.

**Number of nodes on boundaries** If there are boundaries that have 3 or less nodes, the user will be prompted to delete these.

**Node code reset** All nodecodes are reset to: 1 - outer boundary, 2 - island boundary

**Triangulation Check** TQGG will perform a test-triangulation of the current boundaries. If any errors occur, the user will be notified and a permanent marker will be placed on the location where the error occured. Errors include: intersecting boundaries, coincident nodes and more.

#### 4.4.6 Menu item EraseChecks

This option turns off place, vertex and triangle marking.

#### 4.4.7 Menu item PMarkers

Selection of this option allows the user to place a coloured marker anywhere in the current window by means of the mouse. These markers remain displayed until **EraseAll** is invoked. **EraseLast** erases the last marker created. Markers survive windowing and consequently, one of the purposes they can be used for is to identify an area of interest on the grid.

#### 4.4.8 Menu item EraseLast

Selection of this option will delete that last marker that was placed on the grid.

#### 4.4.9 Menu item EraseAll

Selection of this option will delete all markers on the grid.

#### 4.4.10 Menu item SetRange

Selection of this option will allow the user top input a range for the **{Info}TooClose** menu item. The range is in meters.

#### 4.4.11 Menu item TooClose

The TooClose option allows automatic detection of nodes that are too close to one another (coincident). All the nodes in the workspace will be sorted along the x-axis, any node that has another node within its range in the positive x-direction will be flagged to be deleted and marked in the workspace. The range is set using **{Info}SetRange** menu item. If no range has been set when invoking this option, the user will be prompted to input one. The distance between nodes is calculated using the *Haversine* formula for Lat/Lon grids and *Pythagoras' theorem* for meters based coordinates.

After nodes are flagged and marked, the user is prompted to delete them. If confirmed all the flagged nodes will be deleted.

### 4.4.12 Menu item Files

This option brings up a list of files currently assigned to the Editor. Filenames cannot be changed via the display panel under this option.

### 4.4.13 Menu item Limits

This option displays the maximum number of nodes allowed, the number of nodes used at present, maximum neighbours allowed and maximum boundaries allowed.

## 4.5 Contents of menu: GridGen

On picking the GridGen option from {TOP} with the mouse, the options appear as shown in Table 5:

Table 5: GridGen menu items.

| GridGen: |
|---|
| OneFront |
| Clusters<br>Options |
| AllFronts<br>Options |
| OverlayHex<br>OverlaySquares<br>OverlayMixed |
| Triangulate |

These options are used for the creation of nodes and grids. For clusters, the generation is within the working polygon (the currently-activated polygon or the whole polygon if there are no polygons). The last option generates triangles from nodes.

### 4.5.1 Menu item OneFront

One of the options for generating a grid of triangles is a frontal marching method described in [Löhner and Oñate(1998), Sadek(1980)]. The grid generation starts at the boundaries and projects new nodes into the interior based on the location and curvature of the boundary nodes. In order to generate a reasonable grid, the location of the boundary nodes must vary smoothly. This option generates a single front with a line of triangles projected into the interior of the grid. Fronts

can be added one at a time to evaluate how the grid generation is progressing. If a reference grid is input, the triangle area will vary with depth in the reference grid such as the grid becomes coarser with increasing depth.

### 4.5.2 Menu item Clusters

The Clusters option provides a method of creating a set of nodes whose spacing is a function of water depth. As explained in [Henry and Walters(1992)], when the model domain is subdivided into cells whose areas are proportional to water depth as specified in a reference grid and the centres of area of these cells are taken as the basis for a triangular network, the areas of the triangles in the network are also approximately proportional to water depth and the spacing of the nodes is such that the Courant criterion is satisfied approximately throughout the grid, that is, node spacing is proportional to the local phase speed of shallow water waves. It should be noted that triangle area could be made proportional to any scalar quantity defined over the domain, by providing a reference grid for that quantity in place of a depth grid after this menu item is selected.

If the cluster option Display Mesh (below) is chosen, then the fine mesh will be displayed and the user has the option of changing the spacing. Otherwise, 100 cells across is used as a default. After displaying the mesh, the user is queried whether the mesh is OK. If not, the user is prompted for a new number of cells across the grid.

If the cluster option Display Reference Grid (below) is chosen, then the reference grid is displayed. This option is useful for determining whether the reference grid properly overlays the grid being generated.

Next, the model domain is subdivided into cells by forming appropriately-sized compact clusters of squares summed from the fine resolution Cartesian grid laid out over the domain. Cluster (cell) area is related to water depth by an expression A0 + A1*DEPTH + A2*DEPTH**2, where the coefficients A0, A1, A2 can be set by the user. The linear case described in the preceding paragraph corresponds to A0 = A2 = 0. If the display Cluster option (below) is chosen, the user will be queried whether the cluster layout and gradation looks OK. Selecting no will bring up a series of prompts to read in new values of A0, A1, and A2. This is a quick generation procedure where the user is encouraged to experiment until a satisfactory grid is obtained.

### 4.5.3 Menu item Options (Clusters)

When selecting this menu item, the user will be prompted with the following questions:

Display Mesh? Yes or No. If yes, the fine mesh from which clusters are constructed is displayed.

Display Ref Grid? Yes or No. If yes, the reference grid for depth information is displayed.

Display Clusters? Yes or No. If yes, the constructed clusters are displayed.

#### 4.5.4 Menu item AllFronts

This option starts the creation of an unstructured grid composed of triangles using the frontal marching method described above. The frontal method will continue until no additional nodes can be created. The resulting triangle shapes can usually be improved with the reshape option in menu **{GridEdit}Reshape**. If a reference grid is input, the triangle area will vary with depth in the reference grid such as the grid becomes coarser with increasing depth.

#### 4.5.5 Menu item Options (AllFronts)

[options under construction]

#### 4.5.6 Menu item OverlayHex

This option creates a grid of equilateral triangles over an existing grid. First read in a grid file, or a node file and triangulate it, and select **OverlayHex**. The program will prompt then user for the number (or size) of the triangles across the grid. Then a grid will be overlayed on the existing grid and any triangles outside of the outer boundary will be deleted. The resulting grid is then the active grid.

#### 4.5.7 Menu item OverlaySquares

This option creates a grid of square quadrilaterals over an existing grid. First read in a grid file, or a node file and triangulate it, and select **OverlaySquares**. The program will prompt then user for the number (or size) of the squares across the grid. Then a grid will be overlayed on the existing grid and any squares outside of the outer boundary will be deleted. The resulting grid is then the active grid.

#### 4.5.8 Menu item OverlayMixed

This option creates a grid of square quadrilaterals over an existing grid with triangles along the boundary to achieve a better fit. First read in a grid file, or a node file and triangulate it, and select **OverlayMixed**. The program will prompt then user for the number (or size) of the squares across the grid. Then a grid will be overlayed on the existing grid and any squares outside of the outer boundary will be deleted or fitted to a triangle. The resulting grid is then the active grid.

#### 4.5.9 Menu item Triangulate

The following option is used for triangulation of a set of nodes; that is, it converts the data from NODE format to NGH format with a neighbour list and

triangle list. The triangulation algorithm was devised by Prof. Scott Sloan (Department of Civil Engineering, University of Newcastle NSW 2308, Australia). It yields what is known as a constrained Delaunay triangulation, one in which the triangles formed are as near equilateral as possible for the given positions of the nodes.

The algorithm starts by creating a large supertriangle around the domain to be triangulated. Then individual nodes are injected into the triangulation and element edges are swapped as required. Next, boundary and interior line constraints are enforced. Finally, the supertriangle and exterior triangles are removed and the resulting grid is subject to a number of error checks.

After triangulation, the grid should be checked both visually and by using various test operations such as in menu item **{Info}ElementCheck**.

## 4.6 Contents of menu: NodeEdit

Picking EditNode with the mouse brings up a menu with options as shown in Table 6. It gives access to operations affecting individual nodes rather than groups of nodes.

Table 6: NodeEdit menu items.

| NodeEdit: |
|---|
| DeleteNode |
| MoveNode |
| AddBndNode |
| Reverse |
| Join |
| Split |
| ReSample |
| Reselect |
| AddBndLine |
| DeleteIsland |
| AddIntNode |
| AddIntLine |

### 4.6.1 Menu item DeleteNode

This menu option is for deleting single nodes. Point the mouse at the node that is to be deleted, and click.

N.B. This DeleteNode option should not be used for complete deletion of an island, which requires more radical changes to the NODE file. Instead, use DeleteIsland, described later.

#### 4.6.2 Menu item MoveNode

This menu option is for moving single nodes. Point the mouse at the node that is to be moved, and click. The node will be highlighted. Then click on the location where the node should be moved.

#### 4.6.3 Menu item AddBndNode

This menu options is for adding a boundary node to an existing boundary. Click on the location where the node is to be added. TQGG will connect the node to the existing boundary and the user will be prompted to confirm the connection. If the connections cannot be found, insert the node between the desired connections. Then move the node to the proper location.

#### 4.6.4 Menu item Reverse

This option is used to reverse the direction of a boundary. After choosing this option, click on the boundary that is to be reversed.

#### 4.6.5 Menu item Join

This option is used to join two boundary endpoints. After choosing this option, click on the first boundary, then the second boundary endpoint that is to be joined. The user will be prompted to confirm the action.

#### 4.6.6 Menu item Split

This option is used to split a boundary. After choosing this option, click on a boundary where you want to delete it. After doing so, TQGG will suggest splitting the boundary in the direction of the next node that is farthest away from the selected node. If this is rejected, TQGG will suggest splitting in the opposite direction. If confirmed, the boundary will be split. Note that endpoints can not be split from the boundary.

#### 4.6.7 Menu item ReSample

This option performs an uneven ReSample along a boundary line. When selected, the user should select two nodes on the same boundary. TQGG will then perform a resample so that the resolution is evenly increasing/decreasing from the resolution at the first node, to the resolution at the second node.

#### 4.6.8 Menu item Reselect

This option allows the user to replace any designated string of boundary nodes with a fresh selection of nodes from a boundary data file in NODE format. Selection is made on the basis of choosing every Nth digitized point.

#### 4.6.9 Menu item DeleteIsland

This option deletes all boundary nodes associated with an island, that is, complete removal of an island from the node file. The user selects the island to be deleted by picking any boundary node of the island.

#### 4.6.10 Menu item AddIntNode

When this item is selected, the AddInternalNode option is active. Nodes are added in sequence by clicking on the interior of the grid. This option is active until changed by another menu option.

#### 4.6.11 Menu item AddIntLine

This option is used for adding a line of nodes in the interior of the grid. First, the 2 end nodes are selected, then the user is prompted for the number of nodes to place between the endpoints.

### 4.7 Contents of menu: GridEdit

This menu option leads to menu items that provide for manipulation of triangular grids, including editing, merging, and splitting. When the GridEdit menu is chosen, the options in Table 7 appear:

These options permit a wide variety of changes to be made to the displayed grid, as described below.

#### 4.7.1 Menu item AddLine

Option AddLine permits addition of a connection between two vertices. Pick the two nodes by point and click.

#### 4.7.2 Menu item DeleteLine

Option DeleteLine removes of the connection between two vertices. Pick the line that is to be removed by clicking its center point.

#### 4.7.3 Menu item AddNode

Choice of option AddNode permits the user to add a new vertex to the grid. To add a new node - point and click. If a point is inserted into an existing element, connections are made to the element vertices. If the point is outside an existing

Table 7: GridEdit menu items.

| GridEdit: |
|---|
| AddLine |
| DeleteLine |
| AddNode |
| DeleteNode |
| Move |
| Merge |
| CleaveNode |
| Insert |
| Exchange |
| DeKite |
| ReShape |
| GridToNodes |

element, no connections are made and the point must be connected using menu item **AddLine**.

#### 4.7.4 Menu item DeleteNode

This option allows deletion of a vertex and its connections to other vertices. To delete a node - point and click.

#### 4.7.5 Menu item Move

The Move operation consists simply of moving a designated vertex to a new location, which should lie strictly within the polygon formed by the neighbours of the vertex, otherwise some line segments will cross. The vertex and its new location are selected with the mouse and cursor. The display shows the revised positions of the line segments linking the moved vertex to its neighbours, so that the move can be confirmed or cancelled. In the latter case, the vertex and its connections are redisplayed in their original configuration.

#### 4.7.6 Menu item Merge

The Merge operation combines two nodes into one. First, use the mouse to choose the node that should be merged into another, then choose the node that is should be merged into.

Values of depth and computational code are changed automatically when vertices are merged. If the vertex being moved is labelled A, and B designates

the stationary vertex with which A is merged, then the depth and code for the merged vertex at B are set as follows:

- If B and A are both interior points of the grid, B retains its original code (0) and depth.
- If A is an interior point and B is a boundary point, B retains its original code and depth.
- If A is a boundary point and B is an interior point, B assumes the code and depth of A.
- If B and A are both boundary points, B retains its original code and depth. If the points originally lay on boundaries of different types, the user should check whether the code and depth at B should equal the original code and depth at B or A, and reset them with option {EditGroup}Grids.NodeCode if necessary. Depths and codes at individual nodes can be changed if necessary by means of options in {EditGroup}.

N.B. A node can only be merged with one of its neighbours.

#### 4.7.7 Menu item CleaveNode

Cleave allows the user to replace an interior vertex with two new vertices, each of which is connected to roughly half of the neighbours of the vertex being replaced. To carry out a cleaving operation, it is necessary only to use the cursor to select the point to be replaced. The purpose of this option is to allow convenient insertion of extra vertices. Cleaving is not allowed on boundary vertices or with interior vertices having only three or four neighbours. The depth at each new vertex position is computed automatically by linear interpolation. The Reshape option is often used following the cleave operation in order to improve triangle shape.

#### 4.7.8 Menu item Insert

The Insert option is used in two situations, where adding a line connection to the grid implies creation of a new vertex.

First, a line segment can be added from an existing interior node to a point X on the boundary. This requires creation of a new grid node on the boundary, since every line of the grid must end at a node. This operation can be carried out by placing the cursor at or near the mid-point of a boundary segment. If the boundary points on either side of the new boundary vertex have the same computational code, the editor assigns that code to the new vertex also, otherwise the user is asked to enter an appropriate code. The option of setting the depth at the new vertex manually or automatically is then offered. Automatic depth setting means that the new vertex is assigned a depth equal to the average of the depths at the two neighbouring boundary points. If the new vertex lies

outside the former boundary, a yellow marker is placed at the new vertex as a reminder that the reference DEPTH GRID should be updated correspondingly.

The second application of Insert is to add certain extra connections within a quadrilateral consisting of two triangles sharing a common side. This is done by placing the cursor at or near the mid-point of the connection between 2 interior nodes. An extra vertex is added along with new connections. If the user approves the new configuration displayed, a choice of manual or automatic calculation of depth is offered. If automatic evaluation is chosen, the depth at the new vertex is found by linear interpolation if the four surrounding vertices are all interior points of the grid. In the event that one or more of the surrounding vertices is on a boundary, the depth at the new vertex is set equal to the average of the depths at those surrounding points which have non-zero depths.

In both of the uses of Insert described above, the new vertex is placed at the position of the cursor, i.e. close to the mid-point of the existing connection. Its position can be adjusted subsequently using the Move option in {EditGrid} if required.

#### 4.7.9 Menu item Exchange

The function of this editing operation is to swap a diagonal connection in a quadrilateral formed by 2 triangles. In order to perform this process, place the cursor near the mid-point of diagonal line.

#### 4.7.10 Menu item Dekite

The word 'kite' denotes a 4 triangle patch with one node at the center and 4 nodes along the outer edges. This configuration is usually less accurate than other configurations and can lead to local noise in the solution. The kite is identified by a center node with only 4 adjacent nodes. Choosing this menu item will remove kites and replace them with 2 triangles that have the shortest interior edge.

#### 4.7.11 Menu item Reshape

Reshape provides a method for forming more equilateral triangles in the grid by making appropriate adjustments in the positions of interior vertices. Use of this option is recommended after any editing operations that involve adding or deleting any vertices or connections between vertices. Reshape makes three passes through the grid, treats the interior vertices in order of their indices, and leaves a vertex in its original position if the computed adjustment is less than about 1% of the linear dimensions of the polygon formed by its neighbours. The depth at each new vertex position is computed automatically by linear interpolation.

Unlike all the other editing operations, Reshape cannot be reversed; it is recommended that the current grid be saved using {File}InterimSave if there is any likelihood of requiring the grid as it is prior to reshaping.

### 4.7.12 Menu item GridToNodes

Use this option to generate a Node file from the existing grid. This is the inverse operation to triangulation.

## 4.8 Contents of menu: Polygons

This option permits creation, saving, retrieving, activation and deletion of polygonal areas of the model domain in which the user wishes to carry out editing functions accessed subsequently through the Top menu. The menu is is shown in Table 8 :

Table 8: Polygons menu items.

| Polygons: |
|---|
| Create |
| Whole |
| Cycle |
| Delete |
| Read |
| Write |

These options are described next.

### 4.8.1 Menu item Create

The Create option permits design of a new polygon within which to edit nodes. The user picks successive vertices of the required polygon with the cursor, finishing by picking the first vertex a second time to complete the polygon. Once confirmed, the newly designed polygon becomes the active (yellow) polygon (see Activate below).

### 4.8.2 Menu item Whole

This option creates a polygon that includes the entire grid.

### 4.8.3 Menu item Cycle

Repeated picking of option Cycle permits the user to display in sequence all the individual members of the list of stored polygons. Non-active polygons are outlined in red, whereas the active polygon is outlined in yellow.

#### 4.8.4 Menu item Delete

Delete is used in conjunction with Cycle to delete the currently active polygon from the stored list of polygons.

#### 4.8.5 Menu item Read

This option allows the user to read in a named file containing polygons designed and saved during some earlier node-editing session by means of Write. For instance, when running the demonstration case supplied, use Read to read a file named POLYEAST.DAT, which defines a particular polygon to be used if exact comparison with subsequent test outputs to be possible.

#### 4.8.6 Menu item Write

This saves all currently defined polygons to a file named by the user. It may be used in conjunction with Read to input any polygons previously saved.

### 4.9 Contents of menu: NodeInPoly

This option gives access to various editing operations that can be carried out on groups of nodes once one or more working polygons have been set up. If a polygon is not active a message is displayed asking the user to define a polygon first. The menu is as listed in Table 9

Table 9: NodeInPoly menu items.

| NodeInPoly: |
|---|
| ReSample |
| DeleteBnd |
| DeleteInt |
| DeleteAll |

#### 4.9.1 Menu item ReSample

Use this this option to resample all boundary nodes within the current active polygon. The user will be prompted to input a resample distance in meters. TQGG will then resample all boundaries in the active polygon so that they are at that specified resolution.

#### 4.9.2 Menu item DeleteBnd

Use this menu options to delete all boundary nodes within the currently active polygon.

### 4.9.3 Menu item DeleteInt

Use this menu options to delete all internal nodes within the currently active polygon.

### 4.9.4 Menu item DeleteAll

Use this menu options to delete all nodes within the currently active polygon.

## 4.10 Contents of menu: GridInPoly

This option gives access to various editing operations that can be carried out on groups of nodes once one or more working polygons have been set up. If a polygon is not active a message is displayed asking the user to define a polygon first. The menu is as listed in Table 10

Table 10: GridInPoly menu items.

| GridInPoly: |
|---|
| NodeCode |
| ElementCode |
| DeKite |
| ReShape |
| DeleteGrid |
| SplitGrid |
| RefineGrid |
| CutOffGrid |
| SetDepth |
| ReDepth |

### 4.10.1 Menu item NodeCode

This option permits changing all computational codes for nodes within a polygon. For instance, all the nodes with code = 1 along a section of land boundary can be changed to code = 5 for an open boundary. Then the endpoints are set manually to code = 6 to describe a node at the junction of a land and open boundary. These particular codes are used in the model RiCOM but any other codes can be chosen for different models.

### 4.10.2 Menu item ElementCode

This option allows the element code to be changed for groups of elements in the current polygon, or by reading a polygon file. When a grid is set up, separate polygons should be created using **{Polygons}Create** to define separate element types. All these polygons should be saved in a file and then read to set element codes for any modification of this grid. The first polygon (usually the whole polygon) defines element code 1 and the second code 2 etc.

### 4.10.3 Menu item Dekite

Replaces all kites (a 4 triangle patch whose center node has 4 neighbors ) in the active polygon. The operation is the same as that in {EditGrid}Dekite but includes the entire active polygon rather than a single kite.

### 4.10.4 Menu item Reshape

This option allows a reshape of the elements in the current polygon. The operation is similar to that in {EditGrid}ReShape.

### 4.10.5 Menu item DeleteGrid

This option deletes the grid section selected by the current polygon.

### 4.10.6 Menu item SplitGrid

The split option enables division of an existing grid into two separate grids. It is used most frequently to remove surplus parts of a grid outside the open boundaries, after the latter have been positioned. It can also be used to split a large grid into smaller parts, either temporarily, to facilitate editing, or permanently, to provide grids for smaller models. (The opposite process of joining together small grids to make a larger one can be carried out by means of the {File}AddGrid option). In order to use the SplitGrid option, an active ''splitting'' polygon must be created first using DefineGroup. In this case, grid parts inside and outside the polygon are separated into two self-consistently numbered grids.

Before choosing the SplitGrid option, the user must design a polygon that demarcates the intended division. That part of the initial grid, which lies inside the splitting polygon, will eventually be output as an independent grid, and the remaining part outside the polygon will be output as another grid. The inner grid will not contain any nodes outside the polygon nor links (edges) to these nodes. The outer grid will contain the links to the inner grid so that the 2 grids can be joined again.

The user is then led through a series of steps concerning display and output of the two sub-grids produced. When these have been carried out, splitting is complete.

#### 4.10.7 Menu item RefineGrid

When this option is chosen, all triangles within the active polygon are refined by dividing each element at its midside point. Thus one element becomes four. If an element has 2 vertices outside the polygon, it is not refined. If an element has 1 vertex outside the polygon, then the element is refined into 2 elements by connecting the midside of the 2 nodes within the polycon and the node outside the polygon.

#### 4.10.8 Menu item CutOffGrid

Cuts a section of the grid off within the active polygon according to a depth test. The user is prompted for a maximum and minimum value for the z coordinate in the grid. Any nodes outside this range are deleted along with the elements that they are part of.

#### 4.10.9 Menu item SetDepth

This option is used for scaling or offsetting the z coordinate. The user is prompted for a scale factor and an offset. These values are used within the active polygon.

#### 4.10.10 Menu item ReDepth

This option enables reading a depth reference NGH file in order to interpolate new depth values at the grid nodes of the current grid. This function is useful when you have altered the existing grid and wish to specify the correct depths (z coordinate).

### 4.11 Contents of menu: Help

When the item Help is chosen, the options in Table 11 appear.

Table 11: Help menu items.

| Help: |
|---|
| TQGG help |
| About |

#### 4.11.1 Menu item TQGG Help

Refers the user to this documentation in the doc directory.

### 4.11.2 Menu item About

Specifies the revision number of the program. In the Motif version, the information is printed on the terminal.

# 5 Input and output formats

All input routines accept free format input files, i.e. data fields in each record must be separated by at least 1 space character. In general, the first line of the main files is a description of the file contents and the first 4 characters on the line define the file type (see below). Older file formats do not have the file description on the first line and the data may be arranged differently. When saving files, they are written using the latest default format which contains the file description. In this way, legacy formats are updated automatically.

The program will attempt to select the correct file format by first assessing the file delimiter (*.ngh, *.grd, *.nod, or *.xyz,) to determine the general type of file, then by reading the file type at the beginning of the file to determine the format, and finally by parsing the file to find the correct variables. An error message is returned if this selection is unsuccessful.

There are 2 broad categories of files: Grid files that contain nodal and connection information (*.ngh and *.grd) , and Node files that contain only nodal information (*.nod, *.xyz, and *.xsc). Grid and node files are used at different stages in grid generation and can be converted back and forth with the menu item GridGen/Triangulate and GridEdit/Grid2Nodes.

For files that use UTM coordinates, the UTM zone must be specified. There are 2 common formats for the zone; e.g., 10North for zone 10 in the northern hemisphere, or 10U with U specifying the latitude band. In TQGG we use the latter.

## 5.1 Neighbour (NGH) format

### 5.1.1 Description

Neighbour format files can have 3 formats: default (current) format, old format, and really old (original) format. These are defined below. The program will read all 3 formats but only write in the default format.

The default format file can contain comment lines beginning on line 2 and are identified by a # character in the first position on the line. The first line after comments contains scaling and grid type information. When the file is read, the scaling factors scaleX and |scaleZ| are applied to convert to the correct units (degrees or meters). Then scaleX is reset to 1. and scaleZ is reset to sign(scaleZ)*1. The sign of scaleZ then indicates whether the z axis is positive upwards (+1.) or positive downwards (-1.). The offsets from the global coordinate system are not applied but retained in the data set for reference in later calculations.

The old file format originally contained (xmax,ymax,xmin,ymin) instead of ( x_origin, y_origin, scaleX, scaleZ) on the third line. The max/min values are unnecessary and were only used for plotting purposes. They have been replaced with scaling information such as in the default format. IMPORTANT NOTE: Older files with max/min values MUST be updated to contain scaling information or the units will be nonsense.

Since the old and really old formats do not contain a grid type, the user is prompted for this when the file is read. The really old format does not contain scaling information so this must be applied by the user when editing the grid.

**Data layout - default format**:

filetype (string, free format)
x_origin, y_origin, scaleX, scaleZ, igridtype, [UTMzone] ( free format)
np (integer, free format)
nnb (integer, free format)
np lines of
id,x(id),y(id),code(id),z(id),(nbr(id,j),j=1,nnb)

**Data layout - old format**:

np (integer, free format)
nnb (integer, free format)
x_origin, y_origin, scaleX, scaleZ ( free format)
np lines of
id,x(id),y(id),code(id),z(id),(nbr(id,j),j=1,nnb)

**Data layout - original format**:

np (integer, free format)

nnb (integer, free format)
np lines of
id,x(id),y(id),code(id),z(id),(nbr(id,j),j=1,nnb)

**Definitions**:

**filetype** (4 characters) - an indicator of the file type. For ngh files it is #NGH.
**x_origin, y_origin** (real) - the grid origin (0,0) in global coordinates in the x and y dimensions.
**scaleX** (real) - scaling for the grid in the x and y dimensions so that the resultant units are degrees (igridtype=0) or m.
**scaleZ** (real) - scaling of the grid in the z dimension so that the resultant unit is m directed upwards.
**igridtype** (integer) - the type of coordinates used by the grid. 0=(lat,long), 1=UTM, 2=m, 3=general.
**UTMzone** (3 characters) - optional UTM zone if igridtype=1.
**np** (integer) - number of points (nodes) in the grid.
**nnb** (integer) - maximum number of neighbours at a node.
**id** (integer) - index (number) of node , (id = 1, np).
**j** (integer) - neighbour counter, (j=1,nnb).
**x(id),y(id)** (real) - x,y coordinates of id$^{th}$ node.
**code(id)** (integer) - identifies the type of node (boundary,interior,etc).
**z(id)** (real) - the value of bottom elevation at the id$^{th}$ node.
**nbr(j,id)** (integer) - node adjacency array that contains the index of the jth neighbour of the idth node.

### 5.1.2 Example of data file in NGH format

```
#NGH
# Comments go here
0.0000E+00    0.0000E+00    1.0000E+00    -1.0000E+00    3
505
6
1     1.525   12.469   1    0.000     0     2   485     0     0     0
2     1.480   12.280   1    0.000     1     3   400   485   492     0
3     1.310   12.070   1    0.000     2     4   311   477   492     0
4     1.160   11.860   1    0.000     3     5   477     0     0     0
5     1.030   11.640   1    0.000     4     6   381   391   477     0
6     0.910   11.400   1    0.000     5     7   391   482   494     0
7     0.740   11.200   1    0.000     6     8   494   504     0     0
..    .....   .....   .    .....    ..    ..   ...   ...     .     .
..    .....   .....   .    .....    ..    ..   ...   ...     .     .
502    5.249    6.259  0    1.100   122   123   471   486   493     0
503    5.635    6.518  0    1.060   120   121   475   484   496     0
504    0.779   11.055  0    1.030     7     8   401   468   494     0
505    8.486    6.300  0    1.010    68    69   369   457     0     0
```

## 5.2 Node (NOD) format

### 5.2.1 Description

Node format files can have 2 formats: default (current) format and old format. These are defined below. The program will read both formats but only write in the default format.

**Data layout - default format**:

filetype - (string, free format)
x_origin, y_origin, scaleX, scaleZ, igridtype, [UTMzone] ( free format)
np - (integer, free format)
nb,nib - (2 integers, free format)
nbp(1) - (integer, free format)
x,y,z - (nbp(1) lines, 3 reals, free format)
:
nbp(nb) - (integer, free format)
x,y,z - (nbp(nb) lines, 3 reals, free format)
:
npi - (integer, free format)
x,y,z - (npi lines, 3 reals, free format)

**Data layout - old format**:

np - (integer, free format)
nb,nib - (2 integers, free format)
nbp(1) - (integer, free format)
x,y,z - (nbp(1) lines, 3 reals, free format)
:
nbp(nb) - (integer, free format)
x,y,z - (nbp(nb) lines, 3 reals, free format)
:
npi - (integer, free format)
x,y,z - (npi lines, 3 reals, free format)

**Definitions**:

**filetype** (4 characters) - an indicator of the file type. For node files it is #NOD
**x_origin, y_origin** (real) - the grid origin (0,0) in global coordinates in the x and y dimensions.
**scaleX** (real) - scaling for the grid in the x and y dimensions so that the resultant units are degrees (igridtype=0) or m.
**scaleZ** (real) - scaling of the grid in the z dimension so that the resultant unit is m directed upwards.
**igridtype** (integer) - the type of coordinates used by the grid. 0=(lat,long),

1=UTM, 2=m, 3=general.
**UTMzone** (3 characters) - optional UTM zone if igridtype=1.
**np** (integer) - total number of nodes
**nb** (integer) - total number of boundaries (outer, islands, and constraints).
**nib** (integer) - number of internal boundaries (line constraints).
**nbp(i)** (integer) - number of nodes on ith boundary.
**x,y,z** (real)- x,y co-ordinates and bed elevation at node.
**npi** (integer) - number of internal nodes.

NOTES: - outer boundary nodes must be all in one block and must be the first boundary.
- outer boundary must be in counter clockwise order.
- all inner boundaries (islands) must be in clockwise order.
- all internal boundaries that are line constraints are last in the boundary list.
- internal nodes are listed last.

### 5.2.2 Example of data file in NODE format

```
#NOD
# Comments go here
     0.000000  0.000000  1.000000  1.000000  0
     479
       2  0
     146
          1.57         13.47          0.00
          1.46         13.24          0.00
          1.44         13.00          0.00
           :             :              :
          2.70         13.24          0.00
          2.30         13.30          0.00
          1.90         13.40          0.00
      41
          4.09          8.08          0.00
          3.92          8.27          0.00
          3.70          8.44          0.00
           :             :              :
          4.32          7.74          0.00
          4.19          7.97          0.00
     292
          4.29          0.61          5.00
          4.17          1.01          5.00
          4.06          1.41          5.00
           :             :              :
          2.61         11.90         30.00
          2.61         12.30         30.00
          2.66         12.70         30.00
```

## 5.3 Element (ELE) format

### 5.3.1 Description

The element files can have 2 formats: one with 3 integers per line that is suitable for triangle elements only, and one with 5 integers per line that is suitable for quadrilateral or triangular elements. If using the latter, the 4th vertex number is 0 for a triangle. The program will parse the first line to determine the correct format to read. The program will read both formats but only write in the default format.

**Data layout - default format**:

vertex 1, vertex 2, vertex 3, vertex 4, tcode - (5 integers,free format)

**Data layout - triangle format**:

vertex 1, vertex 2, vertex 3 - (3 integers, free format)

**Definitions**:

**vertex 1 to vertex4** - node index for element vertices in CCW order.

### 5.3.2 Example of data file with default element format

```
    1     2    48    0     1
    1    48    41    0     1
    2     3    26    0     1
    2    26    29    0     1
    .     .     .    .     .
    .     .     .    .     .
   32    37    36    0     2
   34    45    35    0     2
   34    35    46    0     1
```

### 5.3.3 Example of data file in triangle only format

```
    1     2    48
    1    48    41
    2     3    26
    .     .     .
    .     .     .
   32    37    36
   34    45    35
   34    35    46
```

## 5.4 Grid (GRD) format

Grid (GRD) format files contain both node location and element vertex information. Node adjacency lists are generated when the file is read. This is different than neighbour files which contain node adjacency information and the element list is generated when they are read.

### 5.4.1 Description

The format for grid (GRD) files has 2 forms. The default format contains a file type identifier and origin and scaling information. The simple format just reads the number of nodes and elements, then reads a list of node locations followed by a list of element vertices. The program will only write in the default format.

**Data layout - default format**:

filetype (string, free format)
x_origin, y_origin, scaleX, scaleZ, igridtype, [UTMzone] ( free format)
np,ne - (2 integers, free format)
x,y,z - (np lines, 3 reals, free format)
OR data layout with node codes:
x,y,z,code - (np lines, 3 reals, 1 integer, free format)
vertex 1, vertex 2, vertex 3, vertex 4, tcode - (ne lines, 5 integers,free format)
OR simple layout for triangles and no Tcodes:
vertex 1, vertex 2, vertex 3 - (ne lines, 3 integers, free format)

**Data layout - simple format**:

np,ne - (2 integer, free format)
x,y,z - (np lines, 3 reals, free format)
OR data layout with node codes:
x,y,z,code - (np lines, 3 reals, 1 integer, free format)
vertex 1, vertex 2, vertex 3, vertex 4, tcode - (ne lines, 5 integers,free format)
OR simple layout for triangles and no Tcodes:
vertex 1, vertex 2, vertex 3 - (ne lines, 3 integers, free format)

**Definitions**:

**filetype** (4 characters) - an indicator of the file type. For grid files it is #GRD
**x_origin, y_origin** (real) - the grid origin (0,0) in global coordinates in the x and y dimensions.
**scaleX** (real) - scaling for the grid in the x and y dimensions so that the resultant units are degrees (igridtype=0) or m.
**scaleZ** (real) - scaling of the grid in the z dimension so that the resultant unit is m directed upwards.
**igridtype** (integer) - the type of coordinates used by the grid. 0=(lat,long),

1=UTM, 2=m, 3=general.
**UTMzone** (3 characters) - optional UTM zone if igridtype=1.
**np** - number of points (nodes) in the grid.
**ne** - number of elements in the grid.
**x,y,z** - coordinates for x,y and bed elevation.
**code** - node code for boundary conditions and other purposes.
**vertex 1 to vertex4** - node index for element vertices in CCW order.
**TCode** - element code for assigning friction type.

### 5.4.2 Example of data file in default GRD format

```
#GRD
# Comments go here
     0.000000  0.000000  1.000000  1.000000  0
          477    795
     1.5700001E+00   1.3470000E+01       0.0000000E+00    1
     1.4299999E+00   1.3180000E+01       0.0000000E+00    1
            .               .                   .           .
            .               .                   .           .
     5.3130002E+00   1.1940000E+00       1.5000000E+01    0
     5.5460000E+00   7.5000000E-01       1.5000000E+01    0
        1         2       204          0    1
        1       204       126          0    1
        2         3       204          0    1
        3         4       203          0    1
        .         .         .          .    .
        .         .         .          .    .
      468       471       472          0    1
      469       470       471          0    1
```

### 5.4.3 Example of data file in simple GRD format

```
          477    795
     1.5700001E+00   1.3470000E+01       0.0000000E+00    1
     1.4299999E+00   1.3180000E+01       0.0000000E+00    1
            .               .                   .           .
            .               .                   .           .
     5.3130002E+00   1.1940000E+00       1.5000000E+01    0
     5.5460000E+00   7.5000000E-01       1.5000000E+01    0
        1         2       204          0    1
        1       204       126          0    1
        2         3       204          0    1
        .         .         .          .    .
        .         .         .          .    .
      468       471       472          0    1
      469       470       471          0    1
```

## 5.5 Node (XYZ) format

This format provides a general method to import nodal data from other datasets such as chart data and shoreline data. It is used in the sampling routines invoked by the menu option File/Sample.

### 5.5.1 Description

The input routines will parse the file to determine the format and variables present, then read to the end of the file to determine the number of data points. Three types of formats are supported. In the first, only the x,y coordinates of the point are read which is primarily used for shoreline data. In the second, x,y coordinates and bed elevation of a point are read which applies to boundary data with depth information. In the third, the data is read as segments in pairs of data points. When the segment identifier is the same and the segments are in order, the segments will be joined together. Otherwise, the segments can be manipulated with editing tools.

**Data layout**:

x,y - (2 reals, free format)
OR with full coordinates
x,y,z - (3 reals, free format)
OR multi-segement with segment number
x,y,z,s - (3 reals, 1 integer, free format)

**Definitions**:

**x,y,z** - x,y coordinates and bed elevation.
**s** - segment number such that all segments where s is the same are joined together.

### 5.5.2 Example of data file in XYZ format

```
 1.5700001E+00   1.3470000E+01      0.0000000E+00
 1.4299999E+00   1.3180000E+01      0.0000000E+00
 1.4800000E+00   1.2880000E+01      0.0000000E+00
 1.5300000E+00   1.2580000E+01      0.0000000E+00
 1.4800000E+00   1.2280000E+01      0.0000000E+00
 1.2700000E+00   1.2020000E+01      0.0000000E+00
 1.0900000E+00   1.1750000E+01      0.0000000E+00
 9.4999999E-01   1.1460000E+01      0.0000000E+00
       .               .                 .
       .               .                 .
```

## 5.6 Section (XSC) format

### 5.6.1 Description

This data format is useful for rivers and other long narrow domains where cross section data at intervals along the thalweg are available. For each cross section, the left bank and right bank coordinates (looking downstream) are specified followed by bed elevations along the cross section.

**Data layout**:

nxp
Next nxp pairs of lines:
ns,xl,yl,xr,yr,zref
( (distr(j), depth(j)) j=1,ns)

**Definitions**:

**nxp** - number of cross-sections to be input.
**ns** - number of points in the input cross-sections.
**xl,yl** - (x,y) coordinates of left bank, looking downstream.
**xr,yr** - (x,y) coordinates of right bank, looking downstream.
**zref** - elevation reference for cross-section.
**distr** - distance from right bank.
**depth** - depth below zref at point distr.

### 5.6.2 Example of data file in XSEC format

```
   21
9  -6.06      3.50      6.06     -3.50      9.15 0.  0.  2.00 1.2 2.50 1.5 3.00 1.8 3.33
2.  4.06 2.  4.80 2.  5.53 2.  6.27 2.
9   3.20     17.48     13.69      8.22      9.145 0.  0.  .77 .45 1.54 .9 2.33 1.35 3.11
1.8 3.88 2.25 4.67 2.25 5.44 2.15 6.22 2.0
9  18.51     28.48     21.72     19.00      9.14 0.  0.  .77 .45 1.54 .9 2.33 1.35 3.11
1.8 3.88 2.25 4.67 2.5 5.44 2.5 6.22 2.05       .         .         .         .         .
.         .         .         .         .         .         .         .         .
9 206.79    -18.15    209.47    -27.64     9.06 0.  0.  .77 .45 1.54 .9 2.33 1.35 3.11
1.39 3.88 1.43 4.67 1.47 5.44 1.51 6.2 1.5
9 214.74    -8.04     225.23    -17.31     9.055 0.  0.  .77 .45 1.54 .9 2.33 1.35 3.11
1.45 3.88 1.55 4.67 1.65 5.44 1.75 6.2 1.8
```

# Acknowledgements

Early development of TRIGRID was carried out at the Institute of Ocean Sciences by Lynda Williams, Kin Wu, and Russ Kirby, under the supervision of Edmand Fok, to specifications developed by Falconer Henry. Modification of the interactive editor to permit colour marking of triangles was done by Robin Parlee, Gordon Laine and John Snider, students in the Camosun College Computer Technology program. The node editor (NODER) was developed by John MacLeod over two workterms of the Camosun College Computing Technology course. Roy Walters, then at U.S. Geological Survey, wrote the sampling program (SAMPLER) and the interactive display and plotting programs PLOTGRID and PLOTMOD, which were later, merged into a single program (DISPLOT).

Reorganisation of the package for conversion to GKS graphics and implementation on a MicroVAX was carried out by Roy Walters. The major task was separating the graphics functions from the computational functions. Daphne Connolly and Steve Prestage of Camosun College later ported the Editor to a PC-AT, at the same time introducing mouse input and other improvements.

Francisco Werner and Mike Hagin, Skidaway Institute of Oceanography, in cooperation with Roy Walters, ported TRIGRID to SUN workstations. Adrian Dolling and John MacLeod, of Channel Consulting, Victoria, ported TRIGRID to the IBM RS 6000 workstation, under contract from IBM Bergen. Dave Greenberg produced the first X-GKS version of TRIGRID and ran it on Sun and Data General workstations and also on a PC with the Linux operating system. Adrian Dolling has ported TRIGRID to MS-Windows for the PC using the WIN API. In addition, two Camosun students, Allan Moore and Sam LeBlanc worked under Adrian Dolling's direction on a graphics interface module for Trigrid in the spring of 1993.

To obtain the present structure of the program, Roy Walters has merged all the graphics programs (including SAMPLER, NODER, EDITOR, and SPLITTER) and DISPLOT. In addition, the functionality in GRIDIT (triangulation) and PREJOIN (grid merging) have been included. The input section was expanded to include grid generation from river cross-sections, and creation of grids in quadrilateral areas.

In 2002, P. Chandramohan (NIWA, New Zealand) collected all the current documentation into a Windows *.hlp file that could be invoked directly from the program menu. With the demise of the programs that display these file formats, the documentation was converted to a pdf file with considerable input from Clayton Hiles.

Recently, Roy Walters has written a new C interface that permits porting to Linux and OSX. The dialog boxes were added by Raymond Rusk. He has also created a GitHub open source repository where the program now resides.

The latest improvements are an expansion and modification to the sampling part of the program by Kristoffer Lorentsen. This work has greatly improved importing geometric data and constructing boundary geometry which then leads to better grids when using frontal marching methods.

# References

[Henry and Walters(1992)] Henry, R.F. and R.A. Walters, 1992. A geometrically-based, automatic generator for irregular triangular networks. Communications in Applied Numerical Methods 9, 555–566.

[Löhner and Oñate(1998)] Löhner, R., Oñate, E., 1998. An advancing front point generation technique. Commun. Numer. Meth. Engng. 14, 1097-1108.

[Sadek(1980)] Sadek, E.A. (1980). A scheme for the automatic generation of triangular finite elements. International Journal of. Numerical Methods in Engineering 15, 1813-1822.

[Thacker(1980)] Thacker, W.C. (1980). A Brief Review of Techniques for Generating Irregular Computational Grids. Int. J. Numerical Methods in Engineering., 15, 1335-41