DRAFT

VOGG: A Visual Orthogonal Grid Generation Tool for Hydrodynamic and Water Quality Modeling

Prepared for:

***US Environmental Protection Agency
Region 4
61 Forsyth St.
Atlanta, GA 30303***

Prepared by:

***Tetra Tech, Inc.
10306 Eaton Place
Suite 340
Fairfax, Virginia 22030***

August 2002

Table of Contents

1. Introduction

The wide application of multi-dimensional hydrodynamic models employing finite difference solution schemes formulated in curvilinear-orthogonal horizontal coordinates has to a certain extent been limited by the difficulty inherent in constructing accurate boundary fitted grids for surface water bodies. Existing grid generation software as exemplified by the GEFDC grid generation program (Hamrick, 1996) generally requires a considerable amount of user artistry and experience to complement the power of the mathematical grid generation algorithm.

The purpose of this report is to provide a complete user's guide for a new visual orthogonal grid generation system, VOGG. The VOGG generation system includes a FORTRAN implementation of a novel physical domain grid generation algorithm, a Windows/GIS based interface for creating necessary input files and displaying output results, and a number of utility programs. The VOGG grid generation system was developed as a component of the US EPA Region 4 TMDL modeling toolbox and specifically supports curvilinear-orthogonal grid generation for the Environmental Fluid Dynamics Code (EFDC). However, a variety of ASCII text output files are created which readily allow grid information to be processed and reformatted for other hydrodynamic and transport models employing both orthogonal and non-orthogonal curvilinear grid formulations.

The remainder of the report is organized as following: Chapter 2 provides an overview of two-dimensional curvilinear grid generation and introduces the primary concepts and terminology. Chapter 3 describes the VOGG Fortran Program including input and output files and command line execution. Chapter 4 describes the use of the AUTOIJ tool in conjunction with the VOGG program. Chapter 5 describes the GRID Windows/GIS based interface mode of grid generation using VOGG and AUTOIJ. Chapter 6 provides a tutorial on using the GRID interface for two sample problems included with the software distribution. A mathematically detailed discussion of two-dimensional curvilinear-orthogonal grid generation in general and the physical plane generation procedure use in VOGG in specific are presented in Appendix A. Appendix B describes two utility programs, FIXSHORE for converting NOAA medium resolution coastal shoreline data in longitudinal-latitude format and FIXBATH for converting NOAA bathymetric sounding and coastal relief data also in longitudinal-latitude format, to formats acceptable to the GRID interface and the VOGG program.

2. Overview of Curvilinear-Orthogonal Grid Generation

The goal of curvilinear-orthogonal grid generation is the construction of boundary fitted model grids to accurately represent geometrically complex regions. In contrast the use of rectangular Cartesian grids requires approximation of boundaries in a jagged or stair-stepped manner. Typically, boundary fitted curvilinear grid are able to represent complex regions with fewer grid cells than Cartesian grids. Mathematically, curvilinear orthogonal grids are required when the model equations are formulated in a curvilinear-orthogonal coordinate system. For example, the two-dimensional Cartesian advection diffusion equation

$$\frac{\partial c}{\partial t}+\frac{\partial}{\partial x}\left(uc\right)+\frac{\partial}{\partial y}\left(vc\right)=\frac{\partial}{\partial x}\left(D\frac{\partial c}{\partial x}\right)+\frac{\partial}{\partial y}\left(D\frac{\partial c}{\partial y}\right) \tag{2.1}$$

has the form

$$h_\xi h_\eta\frac{\partial c}{\partial t}+\frac{\partial}{\partial \xi}\left(h_\eta u_\xi c\right)+\frac{\partial}{\partial \eta}\left(h_\xi u_\eta c\right)=\frac{\partial}{\partial \xi}\left(D\frac{h_\eta}{h_\xi}\frac{\partial c}{\partial \xi}\right)+\frac{\partial}{\partial \eta}\left(D\frac{h_\xi}{h_\eta}\frac{\partial c}{\partial \eta}\right) \tag{2.2}$$

in the two-dimensional curvilinear-orthogonal (ξ,η) coordinate system where u_ξ and u_η are the physical velocity components in the ξ and η directions, and h_ξ and h_η are the metric or scale factors. A common convention in numerical finite difference-finite volume solutions of (2.2), requires the ξ and η coordinates to have integer values equivalent to the two-dimensional array indices (I,J) used to store the discrete variables. When this convention the case, the scale factors, h_ξ and h_η, correspond to the average physical dimensions of the grid cells in the ξ and η directions. Note that (2.2) can also define a rectangular grid cell Cartesian grid when the scale factors are equal the corresponding exact cell dimensions. Thus form a numerical solution implementation perspective, the curvilinear-orthogonal grid generation process is required to provide the scale factor necessary to solve equation (2.2) and also to provide the orientation of the curvilinear coordinate system and velocity vectors relative to fixed geographic coordinates.

The curvilinear-orthogonal grid generation process begins with a definition of the water body region boundary or shoreline as shown for example in Figure 1. Next, the water body boundary is approximated by a curvilinear block rectangular region such as that shown in Figure 2. This block rectangular region should have two important characteristics. First, the sides of the region should be relatively smooth space curves. Second, to ensure a high degree of numerical orthogonality, the curved sides should intersect at right angles at the domain block vertices, shown as bold dots in Figure 2. The VOGG generation scheme and its graphical interface GRID facilitate satisfying these two characteristic, by allowing the user to specify both the (x,y) coordinates of the block vertices, and the slope of the curved line departing from the vertices. When the slope of the departing line is specified, the slope of the incoming line is defined such that the right angle characteristic is satisfied. This allows a curved side to be defined by both the coordinates of its two end points and the line slopes at the two end points. Using this information, the curved sides are represented by smooth cubic Hermite polynomials.

Following the construction of the curvilinear block rectangular approximation of the physical region boundary, the computational domain is defined by a Cartesian block rectangular region having direct correspondence to the sides and vertices of the block curvilinear region. Figure 3 shows a computational domain which corresponds to the physical domain in Figure 2. The integer lengths of each side in the computational domain grid correspond to the number of curvilinear cells along each of the corresponding sides of the physical domain. The dimensioning or length definition of the sides for the computational domain can be done manually or automatically. Manual dimensioning is a trial and error process constrained such that the integer side lengths must result in a closed region or when starting from an arbitrary vertex and moving around the boundary, the starting point should also be the end point. The VOGG grid generation system includes a component AUTOIJ which automatically dimensions the computational domain based on either desired total number of cells in the two index directions or desired physical plane grid cell size in the two curvilinear directions. With the computational domain defined, the grid generation problem can now be viewed as a mathematical mapping of the Cartesian block rectangular computational domain, Figure 3, into the approximate curvilinear block rectangular physical domain, Figure 2. Mathematical methods for constructing this mapping are described in Appendix A. Figure 4 shows the curvilinear grid corresponding to Figures 2 and 3 generated by the VOGG program’s generation or mapping algorithm which described in detail in Appendix A. Also shown in Appendix A is a grid, Figure A11, corresponding to a different dimensioning of the computational domain shown in Figure A10.

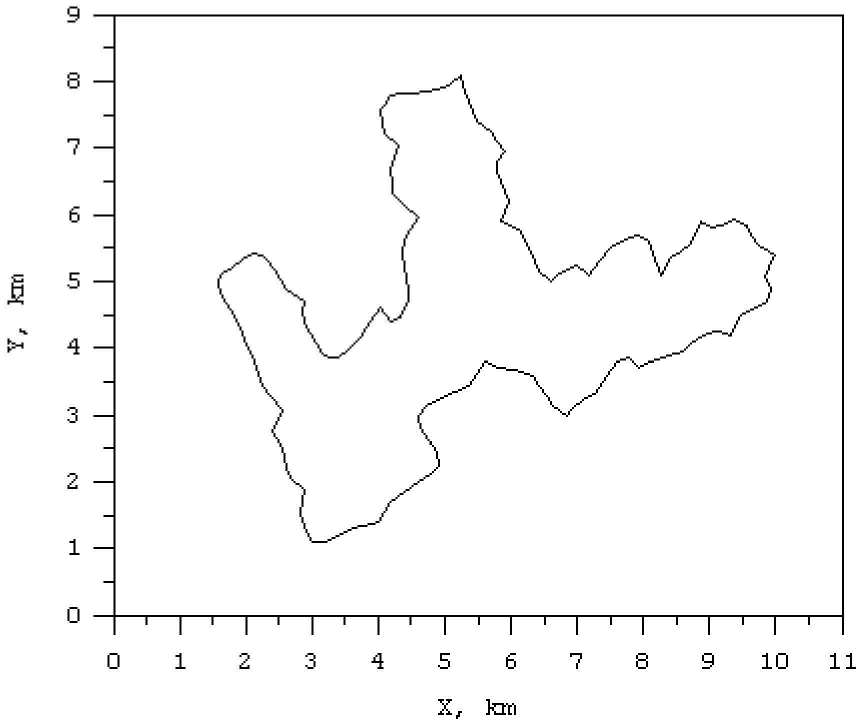


Figure 1. A water body shoreline in physical domain.

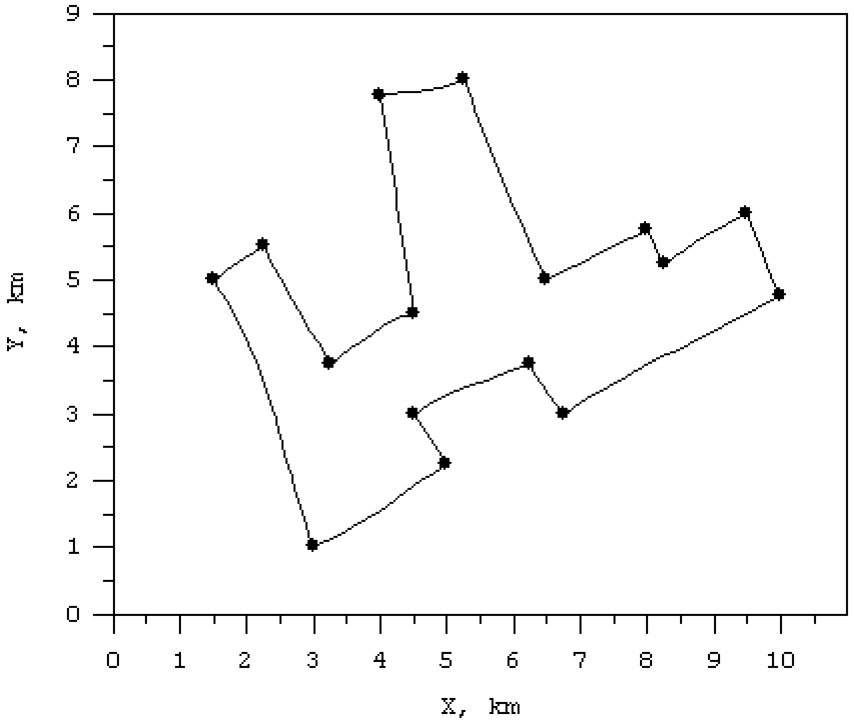


Figure 2. Approximation of water body shoreline in Figure 1 by a curvilinear block rectangular region in the physical domain.

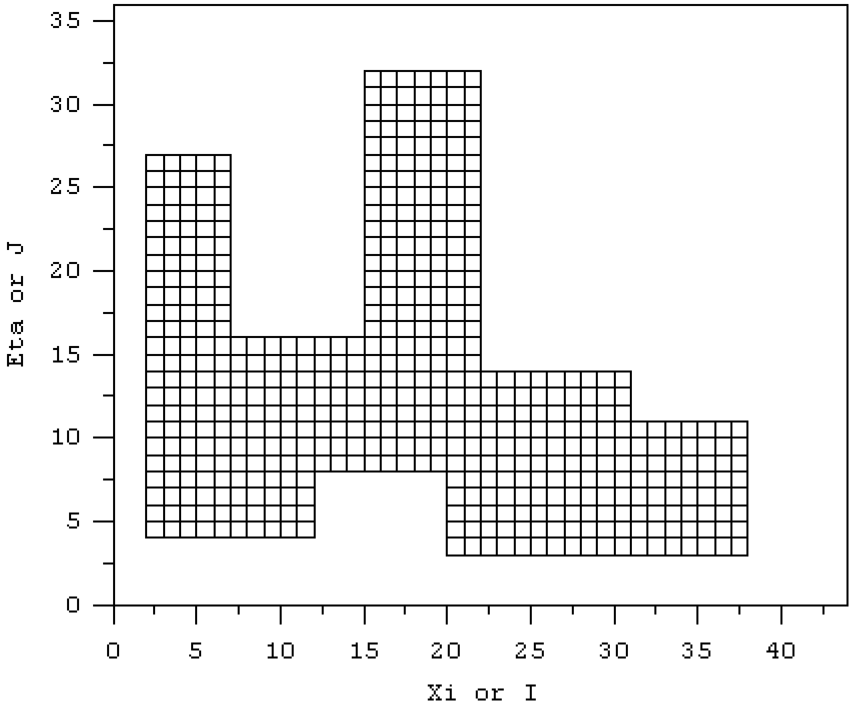


Figure 3. A computational domain corresponding to physical region in Figure 2.

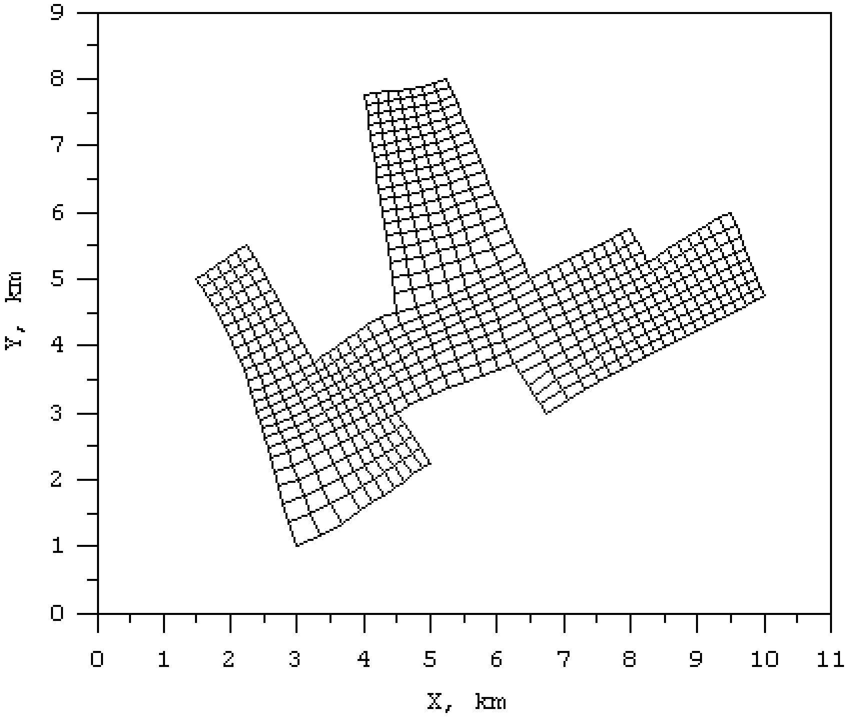


Figure 4. Curvilinear-orthogonal grid corresponding the physical and computational domains shown in Figures 2 and 3, respectively.

3. The VOGG FORTRAN Program and I/O Files

The physical plane grid generation algorithm described in the preceding section and Appendix A is implemented in a FORTRAN program executable vogg.exe obtained by compiling the distributed source file *vogg.f* with the include file *vogg.h*. The VOGG program can be executed directly from the DOS command line, double clicked to execute in a DOS window, or executed within the GRID Window/GIS interface. The VOGG program requires four input files: *cell.inp, gridside.inp, inter.inp,* and *vogg.inp*. If bathymetry or topography are to be interpolated to the grid and included in the output, a file *depdat.inp* is also required. It is recommended that the user gain familiarity with the command line usage of VOGG with example input files before using the GRID interface.

An example of the *cell.inp* file, which defines the computational grid, is shown in Figure 5. The file corresponds to the computational domain grid in Figure 3. The entries in the cell file correspond to a rectangular array of dimension (IC=40, JC=36). Grid cells are identified are defined by three cell type identifiers

0 inactive land cell outside the computational domain

5 active water cell inside the computational domain

9 inactive land cell outside the computational domain but having a side or vertex in common with an active water cell

Note that cell type identifiers in columns 1 and IC must be either type 0 or 9, and that cell type identifies along rows 1 and JC must also be type 0 or 9. The first 4 header lines of the *cell.inp* file are not read. The remaining JC line are read in the format (i3,2x,120i1). For grids having IC exceeding 120, and additional JC row with I beginning at 121 are required. For IC between 241 and 360 additional rows are also required, etc. The *cell.inp* file is also used as an input file for the EFDC hydrodynamic and transport model. After the grid generation is completed, type 5 cells can be changed to types 1, 2, 3, or 4, which define EFDC triangular cells.

The *gridside.inp* file is used to specify information about the grid vertices and sides. Figure 6 shows an example of the *gridside.inp* file corresponding to the physical and computational domains in Figures 2 and 3, and the *cell.inp* file in Figure 5. The first data line *gridside.inp* specifies, nside, the number of vertices or sides, since a block rectangular region will have the same number of vertices as sides. The remainder of the file includes two lines of data for each vertex/side. On the first line of data for each vertex/side the following data fields are specified

nvert = vertex/side identifer number

i_index = I index of the vertex, referenced to lower left corner of grid cell

j_index = J index of the vertex, referenced to lower left corner of grid cell

i_orient = orientation of side
1 - constant I side with land to left (west)
2 - constant I side with land to right (east)
3 - constant J side with land below (south)
4 - constant J side with land above (north)

ijsgn = 1 for curvilinear coordinate increasing along side
2 for curvilinear coordinate decreasing along side

ijcubic = 1 to use interpolation y = cubic function of x
2 to use interpolation x = cubic function of y

ntans = number of tangent points on side (currently must = 0)

The vertex/side numbering convention is that vertex nvert is the beginning point on side nvert. Vertices can be numbered either clockwise or counterclockwise around the perimeter of the block rectangular region. The second line of data for each side includes

x = vertex x coordinate

y = vertex y coordinate

ang = angle in degrees at which the side line departs from the vertex
Measured counter-clockwise from east

i_deflect = 1 for interior corner angle at vertex = 90 degrees, 2 for interior
Corner angle at vertex = 270 degrees

The *gridside.inp* file is read in free format. Note that vertex indices at the start of left (east) sides and above (north) sides are vertices of type nine cells since a cell (I,J) is defined by four vertices (I,J), (I+1,J), (I+1,J+1), and (I,J+1). Also note that the departing side angle, ang, can be adjusted to control the curvature and shape of the side line.

The files *inter.inp* is used to specify grid line interpolation levels. An example is shown in Figure 7, which corresponds to the *cell.inp* and *gridside.inp* example files. The first line of the file includes:

intertype = type of grid line interpolation (must set to 1)

nilevel = number of I or ξ interpolation levels

njlevel = number of J or η interpolation levels

idum = 2

jdum = 2

The first line is followed by nilevel lines of I level data having two fields

idum = I value corresponding to constant I lines in computational grid

rival = value to be interpolated. Default is REAL (idum)

These lines are followed by njlevel lines of J level data having two fields

jdum = J value corresponding to constant J lines in computational grid

rjval = value to be interpolated. Default is REAL (jdum)

For the computational domain in Figure 3 and the corresponding cell.inp file in Figure 5, the I grid lines range from 2 to 39 which defines nilevels as 38. Likewise, the J grid lines range from 2 to 33 which defines njlevels as 32. For values of idum equal to I's on constant I sides, rival must equal idum. Likewise for values of jdum equal equal to J's on constant J sides rjval must equal jdum. Otherwise rival and rjval do not have to be set to the default values.

The file *vogg.inp* is the master control file for the vogg.exe program. An example, compatible with preceding example files is shown in Figure 8. The file has 3 none data header lines each preceding one of the three data lines. The 12 fields on the first data line are default values and should not be changed. The second data line has three fields

ic = number of I cells (number of columns) in *cell.inp* file

jc = number of J cells (number of rows) in *cell.inp* file

cvtdxy = conversion factor between physical domain or vertex coordinates and
Grid cell scale factors.

The EFDC model requires input of grid cell scale factors (grid cell dimensions) in meter. If the grid vertex coordinates in gridside.inp are in kilometer, for example, then cvtdxy should be set to 0.001. The third data line controls the interpolation of bathymetry or topography, provided in the *depdat.inp* file to the generated grid. The fields in this data line are:

isdep= 0 no depth interpolation
1 data in depdat.inp is postive depth below 0 datum
2 data in depdat.inp is topographic elevation

cvtdepa= additive conversion factor for depth data
converted = cvtdepm*(depth+cvtdepa) = units of meters

cvtdepm= multiplying conversion factor for depth data
converted = cvtdepm*(depth+cvtdepa) = units of meters

isdpt= 1 for radius of influence interpolation
2 for points in cell interpolation

surfel= water surface elevation initial water surface elevation
(should be 0.0 for isdep=1)

depmin= minimum initial depth

cdep= radius of influence interpolation power 1. or 2.

radm= radius multiplier 0.25 to 2.0

isfill= fill depths in cells having no nearby data

The isdep parmeter controls interpolation of depth and bottom elevation to the grid. When set to 0, the initial depth is set to depmin and the bottom elevation is set to the negative of depmin. Setting isdep equal to 1 is general used for coastal applications and the data in depdat.inp should correspond to positive depths below mean sea level after the conversions are applied. Bottom elevations are then set to the negative of depth. Setting isdep equal to 2 is generally used for inland applications and the data in depdat.inp should correspond to bottom elevations after the conversions are applied. Initial depths are then determined as the difference between the initial water surface elevation, surfel, and the interpolated bottom elevation. The parameter isdpt determines the type of interpolation used to interpolate data from the *depdat.inp* file to the center of the grid cells. For isdpt equal to 1, a radius of influence interpolation is used as follows on a cell by cell basis. First a minimum and a maximum radius are determined for each cell

Rmin = radm*min(h_ξ,h_η)

Rmax = radm*max(h_ξ,h_η)

The converted data in the depdat.inp are then searched and a distance from the cell center to the data location, Rdata is determined. If Rdata is less than Rmin, the data is used and a weight is calculated

Wt = (Rmin+Rdata)**cdep/(Rmin-Rdata)**cdep

The resulting depth for the cell is then the weighted average of all data within the radius Rmin of the cell center. If no data fall within this radius, the process is repeated using Rmax instead of Rmin. If no data fall within this radius, the cell is tagged as having no nearby data. For isdpt equal to 2, the depth is set to the simple average of all data falling within the cell, with the cell tagged if no data falls within the cell. Setting isfill equal to 1 fills depth and bottom elevation to cells having no nearby data in the interpolation process. The fill interpolates values determined for cells having nearby data to cell identified as having no nearby data.

The file *depdat.inp* contains data for interpolating depth and bottom elevation to the generated grid. The file can have up to 100,000 lines of data with each line having three fields.

X = x or east coordinate of data location

Y = y or north coordinate of data location

Z = data value

The x and y coordinates should be relative to the same coordinate system as the vertex coordinates in gridside.inp. The data values after conversion should have meter units. If the isdep parameter is gridside.inp is set to 1 or 2, and the *depdat.inp* file is absent or missing, the VOGG program resets isdep to 0. If more than 100000 data triples are needed, the value of the parameter ndepm in the include file *vogg.h* should be appropriately increased and the *vogg.f* source file recompiled.

Successful execution of the VOGG program creates numerous output files, many of which are used for diagnostic purposes. The output files include:

Bndry.out – diagnostic of boundary condition implementaion

Dxdy.inp – EFDC input file with cell dimensions, depths, etc.

Efdccell.inp – cleaned version of cell.inp

Efdcgrid.inp – grid definition file for EFDC Explorer Interface

Grid.dpt – grid plotting file in DATAPLOT pen lift format

Gridext.gao – finite element working grid file

Gridext.out – grid definition file in GEFDC output format

Gridext.phy – same as gridext.ou

Gridgao.dia – diagnostic file for finite element grid

Gridgao.dpt – finite element grid plotting file in DATAPLOT format

Gridgao.int – initial grid file for diagnostics

Gridgao.out – finite element grid definition file in GEFDC output format

Gridij.out – vertex diagnostic file, I ordered

Gridji.out – vertex diagnostic file, J ordered

Gridside.dia – diagnositics of gridside.inp processing

Gridside.out – plotting file of cubic side interpolation

Invert.dia – diagnostic file

Invertfull.dia – diagnostic file

Lxly.inp – EFDC input file with cell coordinates and orientation

Salt.inp – EFDC salt.inp template file

Temp.inp – EFDC temp.inp template file

Vogg_grid.plt – grid plotting file in TECPLOT format

Vogg_phys.plt – finite element grid in TECPLOT format

Vogg_physij.out – I vertex interpolation diagnostics

Vogg_physji.out – J vertex interpolation diagnostics

Voggdia.out – grid generation solver diagnostics

The generated grid can be view using the *grid.dpt* or *vogg_grid.plt* files. The grid.dpt file is a 3 column ASCII text file. For each water cell, five lines are written

```
x(i,j)          y(i,j)          -1
x(i+1,j) y(i+1,j) 1
x(i+1,j+1)      y(i+1,j+1)      1
x(i,j+1) y(i+1,j+1)      1
x(i,j)          y(i,j)          1
```

which define a closed polyline of the cell. The third column is a pen lift control with –1 indicating the start of a new polyline. Many x-y plotting programs including DATAPLOT support this format, which is also easily converted to a GIS shape file format. The vogg_grid.plt file is in the standard format for TECPLOT, a widely used commercial engineering and scientific data visualization package. Information is the *gridext.out* or *efdcgrid.inp* files provide sufficient grid vertex information to define grids for most hydrodynamic and transport models using horizontal curvilinear coordinates.

The *efdccell.inp* file is a cleaned version of the *cell.inp*, which corrects potential mistakes in define type nine cells. It can be renamed to *cell.inp* for use with the EFDC model. The files dxdy.inp and lxly.inp are standard EFDC grid information files. Likewise, the *salt.inp* and *temp.inp* are EFDC initial salinity and temperature distribution files containing uniform default data.

```
C    cell.inp file, I across, J down
C             1         2         3         4
C    1234567890123456789012345678901234567890
C
 34  0000000000000000000000000000000000000000
 33  0000000000000099999999000000000000000000
 32  0000000000000095555559000000000000000000
 31  0000000000000095555559000000000000000000
 30  0000000000000095555559000000000000000000
 29  0000000000000095555559000000000000000000
 28  0000000000000095555559000000000000000000
 27  0000000000000095555559000000000000000000
 26  9999990000000095555559000000000000000000
 25  9555590000000095555559000000000000000000
 24  9555590000000095555559000000000000000000
 23  9555590000000095555559000000000000000000
 22  9555590000000095555559000000000000000000
 21  9555590000000095555559000000000000000000
 20  9555590000000095555559000000000000000000
 19  9555590000000095555559000000000000000000
 18  9555590000000095555559000000000000000000
 17  9555590000000095555559000000000000000000
 16  9555590000000095555559000000000000000000
 15  9555599999999955555559000000000000000000
 14  9555555555555555555559000000000000000000
 13  9555555555555555555559999999990000000000
 12  9555555555555555555555555555590000000000
 11  9555555555555555555555555555590000000000
 10  9555555555555555555555555555599999999990
  9  9555555555555555555555555555555555555590
  8  9555555555555555555555555555555555555590
  7  9555555555555555555555555555555555555590
  6  9555555555599999999555555555555555555590
  5  9555555555590000009555555555555555555590
  4  9555555555590000009555555555555555555590
  3  9555555555590000009555555555555555555590
  2  9999999999990000009555555555555555555590
  1  0000000000000000009999999999999999999990
C             1         2         3         4
C    1234567890123456789012345678901234567890
C
```

Figure 5. Example of the *cell.inp* file.

```
   16  ! number of vertexs/sides
 1    2     3      3     1     1    0
                                3.0   1.0     20.     1
 2   12     3      2     1     2    0
                                5.0   2.25   122.     1
 3   12     7      3     1     1    0
                                4.5   3.0     33.     2
 4   20     7      1    -1     2    0
                                6.25  3.75   -67.     2
 5   20     2      3     1     1    0
                                6.75  3.0     33.     1
 6   39     2      2     1     2    0
                               10.0   4.75   119.     1
 7   39    10      4    -1     1    0
                                9.5   6.0   -157.     1
 8   31    10      2     1     2    0
                                8.25  5.25   121.     2
 9   31    13      4    -1     1    0
                                8.0   5.75  -154.     1
10   22    13      2     1     2    0
                                6.5   5.0    116.     2
11   22    33      4    -1     1    0
                                5.25  8.0   -158.     1
12   15    33      1    -1     2    0
                                4.0   7.75   -80.     1
13   15    15      4    -1     1    0
                                4.5   4.5   -171.     2
14    7    15      2     1     2    0
                                3.25  3.75   121.     2
15    7    26      4    -1     1    0
                                2.25  5.5   -151.     1
16    2    26      1    -1     2    0
                                1.5   5.0    -56.     1

! 1st line of vertex/side data
! nvert = vertex/side identifer number
! i_index = I index of the vertex
! j_index = J index of the vertex
! i_orient = orientation of side, 1 - constant I side with land to left (west), 2 –
!   constant I side with land to right (east), 3 - constant J side with land below
!  (south) 4 - constant J side with land above (north)
! ijsgn = 1 for curvilinear coordinate increasing along side, 2 for curvilinear
!   coordinate decreasing along side
! i_orient = orientation of side. 1  - constant I side with land to left (west). 2  -
!   constant I side with land to right (east), 3  - constant J side with land below
!   (south), 4  - constant J side with land above (north)
! ijsgn = 1 for curvilinear coordinate increasing along side
!   2 for curvilinear coordinate decreasing along side
! ijcubic = 1 to use interpolation y = cubic function of x, 2 to use interpolation
!   x = cubic function of y
! ntans = number of tangent points on side (currently must = 0)
! 2nd line of vertex side data
!  x = vertex x coordinate
!  y = vertex y coordinate
!  ang =  angle in degrees at which the side line departs from the vertex , counter-
!    clockwise from east
!  i_deflect  = 1 for interior corner angle at vertex = 90 degrees,  2 for interior
!    corner angle at vertex = 270 degrees
```

Figure 6. Example of the *gridside.inp* file.

```
1    38    32   2    2
  2     2.
  3     3.
  4     4.
  5     5.
  6     6.
  7     7.
  8     8.
  9     9.
 10    10.
 11    11.
 12    12.
 13    13.
 14    14.
 15    15.
 16    16.
 17    17.
 18    18.
 19    19.
 20    20.
 21    21.
 22    22.
 23    23.
 24    24.
 25    25.
 26    26.
 27    27.
 28    28.
 29    29.
 30    30.
 31    31.
 32    32.
 33    33.
 34    34.
 35    35.
 36    36.
 37    37.
 38    38.
 39    39.
```

Figure 7. Example of the *inter.inp* file (continued on next page).

```
     2     2.
     3     3.
     4     4.
     5     5.
     6     6.
     7     7.
     8     8.
     9     9.
    10    10.
    11    11.
    12    12.
    13    13.
    14    14.
    15    15.
    16    16.
    17    17.
    18    18.
    19    19.
    20    20.
    21    21.
    22    22.
    23    23.
    24    24.
    25    25.
    26    26.
    27    27.
    28    28.
    29    29.
    30    30.
    31    31.
    32    32.
    33    33.


! 1 header line - intertype   nilevel  njlevel  idum jdum

! followed by nilevel lines of i level data - idum ival

! followed by njlevel lines of j level data - jdum jval

! nilevel runs from the lowest i vertex index to the highest  default is unit spacing

! njlevel runs from the lowest j vertex index to the highest  default is unit spacing
```

Figure 7. Example of the *inter.inp* file (continued from previous page).

```
C itype ijinter isolv ivert innerm iouterm rpar  resdm    errm   itern  errn   rpgao
  1     1       2     2     2000   800     1.8   1.e-12   1.e-3  1000   1.E-8   1.0
C ic   jc   cvtdxy
  40   34   0.001
C isdep   cvtdepa   cvtdepm  isdpt   surfel  depmin   cdep   radm  isfill
  2       0.0       1.0      2       0.0      0.01    1.0    0.5    1


! itype 0 read external working grid from gridext.inp
!         and boundary conditions from bndry.inp (this was used for initial testing)
!       1 normal operation
!       2 stops after creating fill or initial grid (good place to adjust control
!          points and number of cells per side before a itype=1 run
! ijinter 0 automatically sets real i,j levels for final grid vertex interpolation
!         1 reads real i,j levels for final grid vertex interpolation
! isolv 2 use this value by default (solver type) isolv=1 is inferior
! ivert 2 use this value by default (vertex interpolation control) ivert=1 is inferior
! innerm  max number of inner interations 2000 is ok
! ioutern max number of outer interations 800 is ok
! rpar=1.8 default relaxation solution parameter
! resdm=1.e-12 equation solver convergence this value is ok
! errm=1.e-3 vertex location error this value is ok
! itern = max interpolation iterations this value is ok
! errn = interpolation error criteria this value is ok
! rapgao= relaxation paramter for ao grid
! ic= number i direction cells in cell.inp
! jc= number j direction cells in cell.inp
! cvtdxy= conversion form shoreline units to cell dx and dy units
!         (0.001 for km to meters)
! isdep= 0 no depth interpolation
!        1 data in depdat.inp is postive depth below 0 datum
!         2 data in deptat.inp is topographic elevation
! cvtdepa= conversion factor for depth data
! cvtdepm= conversion factor for depth data
!         converted = cvtdepm*(depth+cvtdepa)
! isdpt= 1 for radius of influence interpolation
!        2 for points in cell interpolation
! surfel= water surface elevation for isdep=2
! depmin= minimum initial depth
! cdep=  radius of influence interpolation power 1. or 2.
! radm=  radius multiplier  0.25 to 2.0
! isfill= fill depths in cells having no nearby data
```

Figure 8. Example of the *vogg.inp* file.

4. The AUTOIJ Fortran Program and I/O Files

The utility program AUTOIJ can be used to automatically set the vertex I and J indices in the *gridside.inp* file. The source file *autoij.f* and the include file *vogg.h* are complied to create the executable **autoij.exe** which can be run for the DOS command line, double clicked to execute in a DOS window or in the GRID interface. AUTOIJ requires two input files *autoij.inp* and *gridside.nij*. The autoij.inp file has one line of data with 6 fields:

itype = 0 to base I,J setting on icside and jcside
1 to base I,J setting on dxauto and dyauto

icside = desired maximum I vertex index

jcside = desired maximum J vertex index

dxauto = desired grid size in the I or ξ direction (h_ξ)

dyauto = desired grid size in the J or η direction (h_η)

icwc = 1 for clockwise ordering to grid points
2 for counterclockwise ordering of grid points

The choice of itype determines the target mode for I,J setting of the vertices based either on the range of I and J or the desired grid resolution in the curvilinear coordinate directions. Generally itype equal to 1 is preferred. For undistorted grid cells, dxauto and dyauto should be equal or of similar magnitude. The units for dxauto and dyauto should be the same as those used for the vertex coordinates. The method employed for setting the vertex I,J indices requires knowledge of weather the vertex point are order clockwise or counterclockwise around the boundary of the physical domain. An example *autoij.inp* file consistent with the files in Chapter 3 is shown in Figure 9.

The file gridside.nij has the same format at the gridside.inp file. An example *gridside.nij* file corresponding to the curvilinear block rectangular shoreline approximation in Figure 2 is shown in Figure 10. The file differs from the standard gridside.inp file in that the indices I_index and j_index are set to zero. The parameter ijsgn is also set to zero. Upon execution, AUTOIJ creates the output file *gridside.inp*. When working in the command line mode, the user must construct a *cell.inp* file corresponding to the created *gridside.inp* file. The *gridside.inp* created by AUTOIJ is shown in Figure 11 and the constructed cell.inp file is shown in Figure 12. Figures A11 and A12 show the computational domain and the generated grid corresponding to these files.

```
  1    0   0   0.2   0.2   -1

! i    i     j   d    d     i
! t    c     c   x    y     c
! y    s     s   a    a     w
! p    i     i   u    u     c
! e    d     d   t    t
!      e     e   o    o
!
! itype=0 use specified  icside, jcside
!      =1 use dxauto, dyauto
!
! iccwc = 1 clockwise ordering
!        -1 counter closckwise ordering
```

Figure 9. Example of the *autoij.inp* file.

```
   16
 1    0     0      3     0     1    0
                                 3.0   1.0     20.     1
 2    0     0      2     0     2    0
                                 5.0   2.25   122.     1
 3    0     0      3     0     1    0
                                 4.5   3.0     33.     2
 4    0     0      1     0     2    0
                                 6.25  3.75   -67.     2
 5    0     0      3     0     1    0
                                 6.75  3.0     33.     1
 6    0     0      2     0     2    0
                                10.0   4.75   119.     1
 7    0     0      4     0     1    0
                                 9.5   6.0   -157.     1
 8    0     0      2     0     2    0
                                 8.25  5.25   121.     2
 9    0     0      4     0     1    0
                                 8.0   5.75  -154.     1
10    0     0      2     0     2    0
                                 6.5   5.0    116.     2
11    0     0      4     0     1    0
                                 5.25  8.0   -158.     1
12    0     0      1     0     2    0
                                 4.0   7.75   -80.     1
13    0     0      4     0     1    0
                                 4.5   4.5   -171.     2
14    0     0      2     0     2    0
                                 3.25  3.75   121.     2
15    0     0      4     0     1    0
                                 2.25  5.5   -151.     1
16    0     0      1     0     2    0
                                 1.5   5.0    -56.     1

!  notes
!
!  i_index, j_index, and ijsgn are set to zero and with be determined
!  by the autoij program
```

Figure 10. Example of the *gridside.nij* file.

```
 16
   1     2     2     3     1     1     0
       3.000         1.000         20.000       1
   2    13     2     2     1     2     0
       5.000         2.250        122.000       1
   3    13     8     3     1     1     0
       4.500         3.000         33.000       2
   4    22     8     1    -1     2     0
       6.250         3.750        -67.000       2
   5    22     4     3     1     1     0
       6.750         3.000         33.000       1
   6    40     4     2     1     2     0
      10.000         4.750        119.000       1
   7    40    12     4    -1     1     0
       9.500         6.000       -157.000       1
   8    32    12     2     1     2     0
       8.250         5.250        121.000       2
   9    32    15     4    -1     1     0
       8.000         5.750       -154.000       1
  10    23    15     2     1     2     0
       6.500         5.000        116.000       2
  11    23    31     4    -1     1     0
       5.250         8.000       -158.000       1
  12    16    31     1    -1     2     0
       4.000         7.750        -80.000       1
  13    16    14     4    -1     1     0
       4.500         4.500       -171.000       2
  14     8    14     2     1     2     0
       3.250         3.750        121.000       2
  15     8    24     4    -1     1     0
       2.250         5.500       -151.000       1
  16     2    24     1    -1     2     0
       1.500         5.000        -56.000       1
```

Figure 11. The *gridside.inp* file created by AUTOIJ for *gridside.nij* file in Figure 10.

```
C cell.inp file, I across, J down
C             1         2         3         4
C    1234567890123456789012345678901234567890
C
 34  0000000000000000000000000000000000000000
 33  0000000000000000000000000000000000000000
 32  0000000000000000000000000000000000000000
 31  0000000000000099999999000000000000000000
 30  0000000000000095555559000000000000000000
 29  0000000000000095555559000000000000000000
 28  0000000000000095555559000000000000000000
 27  0000000000000095555559000000000000000000
 26  0000000000000095555559000000000000000000
 25  0000000000000095555559000000000000000000
 24  9999999000000095555559000000000000000000
 23  9555559000000095555559000000000000000000
 22  9555559000000095555559000000000000000000
 21  9555559000000095555559000000000000000000
 20  9555559000000095555559000000000000000000
 19  9555559000000095555559000000000000000000
 18  9555559000000095555559000000000000000000
 17  9555559000000095555559000000000000000000
 16  9555559000000095555559000000000000000000
 15  9555559000000095555559999999999990000000
 14  9555559999999995555555555555555590000000
 13  9555555555555555555555555555555590000000
 12  9555555555555555555555555555555599999999
 11  9555555555555555555555555555555555555559
 10  9555555555555555555555555555555555555559
  9  9555555555555555555555555555555555555559
  8  9555555555555555555555555555555555555559
  7  9555555555559999999995555555555555555559
  6  9555555555559000000095555555555555555559
  5  9555555555559000000095555555555555555559
  4  9555555555559000000095555555555555555559
  3  9555555555559000000099999999999999999999
  2  9555555555559000000000000000000000000000
  1  9999999999999000000000000000000000000000
C             1         2         3         4
C    1234567890123456789012345678901234567890
C
```

Figure 12. The cell.inp file corresponding to the gridside.inp file in Figure 11.

5. *VOGG Grid Generator Interface*

5.1 Introduction

The VOGG Grid Generator provides a user-friendly interface for creating a curvilinear orthogonal grid for use with the EFDC model. The interface accepts georeferenced data so that GIS data (e.g. shapefiles, coverages) can be used as a template for grid domain assignment. Shoreline, bathymetry, and other spatial data that may assist in creating a conceptual domain may be used as templates. The grid produced by the VOGG Grid Generator interface and results of the EFDC simulation are thus georeferenced and can be integrated into an existing GIS project.

5.2 Left Tool Bar

Figure 1 shows the VOGG Grid Generator interface and describes the buttons in the left tool bar. This tool bar includes all buttons needed for populating the input files required for generating a grid using VOGG (see Sections 3 and 4 for input file structure).

Line Editing

The *Line Editing Tools* allow the user to sketch the domain boundaries. These lines serve as a guide for placing control points and are not used by the VOGG program.

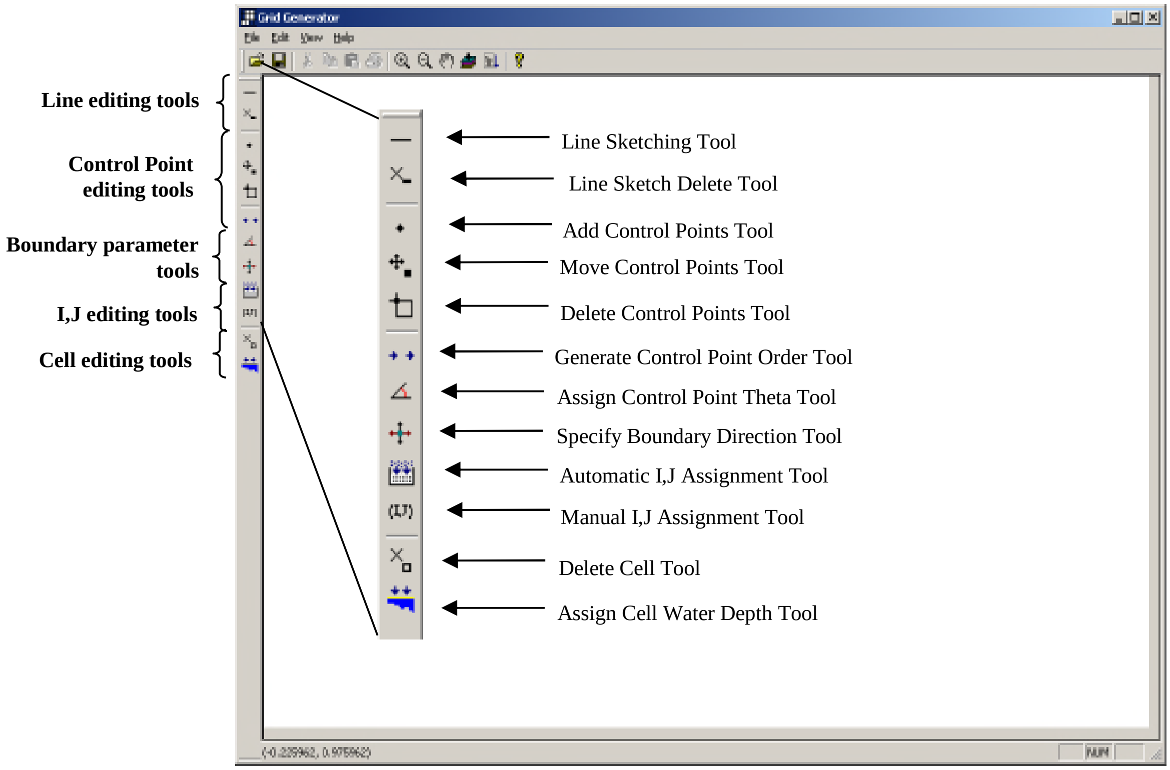


Figure 1. The left tool bar and VOGG Grid Generator Interface.

Control Point Editing

The *Control Point Editing Tools* allow the user to indicate locations from which the mesh configuration can be calculated. If the sketching tool was used to delineate the grid boundary, the Control Points do not have to be placed exactly on these lines. These points are assigned in a clockwise order to form the boundary of the domain. The order must be assigned in a clockwise assignment pattern beginning with a value of 1 (see Tutorial).

Boundary Parameter Editing

Additional information must be assigned to the control points in order to define the mesh properties. Each point is assigned a default value for θ (theta), which can be edited by the user by selecting the *Assign Control Point Theta Tool*. The value of θ for a given node represents the angle measured from due east (+ number = ^{0}counterclockwise from east, - number= ^{0}clockwise from east) of the boundary segment in the clockwise direction of the node.

The *Specify Boundary Direction Tool* is used to assign the orientation of mesh boundaries. Keep in mind that two adjacent nodes may not have the same orientation. After the grid has been oriented, conceptual coordinates (I,J) must be assigned to the control points to construct the computational grid.

I,J Editing

Conceptual I,J coordinates must be assigned to the control points so that a computational grid can be developed. These coordinates must be assigned using either the *Automatic I,J Assignment Tool* or the *Manual I,J Assignment Tool*. The *Automatic I,J Assignment Tool* provides the user with two methods of assigning I,J coordinates. The user may indicate the length of the cells on the x and y axes using the template coordinate system units, or assign the number of cells on the x and y axes. The *Manual I,J Assignment Tool* allows the user to define the I,J coordinates point by point.

Cell Editing

The *Cell Editing Tools* allow for the manipulation of grid cells after the grid has been generated. The *Delete Cell Tool* allows the user to remove cells that overlay land segments such as peninsulas or islands. This feature allows some flexability when assigning control points, enabling the user to grid the entire study area and remove outlying cells at a later point. The user may use buttons in the top tool bar to identify discrete areas to shape using this method.

5.3 Upper Tool Bar

The upper tool bar (Figure 2) in the VOGG Grid Generator Interface also contains useful tools for grid creation. Four of these buttons are used for manipulating the display, and have icons similar to their counterparts in other GIS applications. These include two zooming buttons, a panning button, and a zoom to maximum extent button to assist in viewing at appropriate scales.

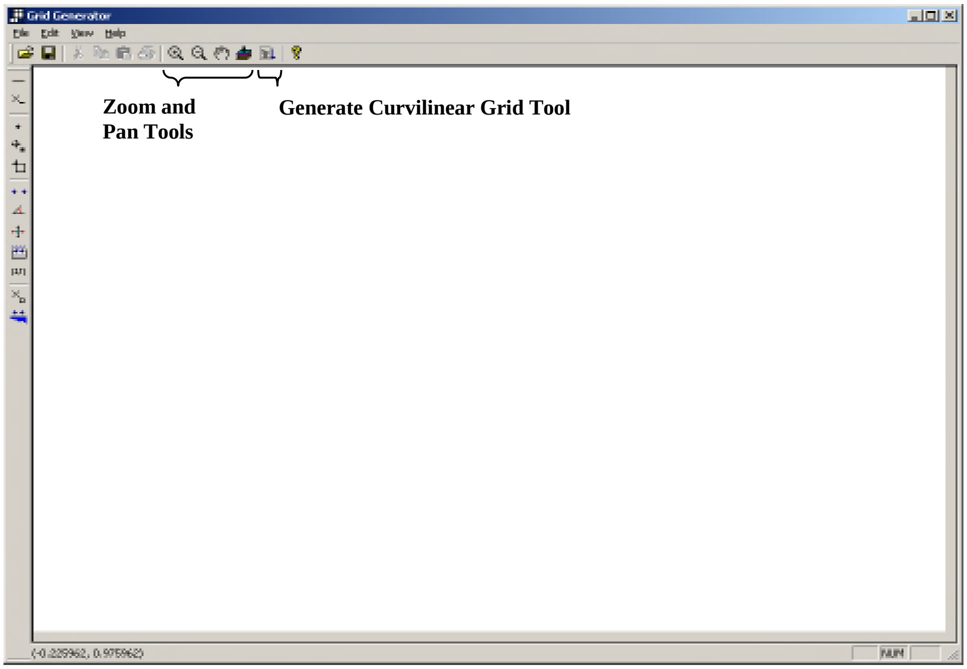
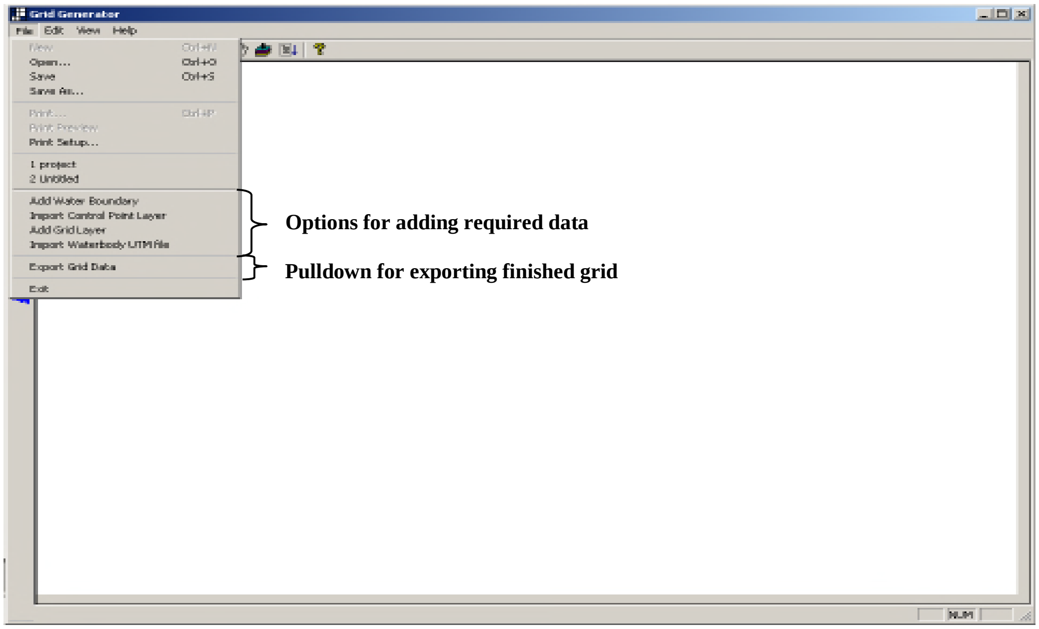


Figure 2. The Top Tool Bar of the VOGG Grid Generator Interface.

The *Generate Curvilinear Grid* button also resides on the upper tool bar. This button initiates the Fortran VOGG program after the control points have been assembled using the tools on the left tool bar. This button is also used to import the grid as a shape file once it has been produced by VOGG.

5.4 Pulldown Functions

Functions specifically designed for the VOGG Grid Generator Interface have also been added as pulldown menus located at the top of the GUI. These pulldowns include functions that allow the user to add or remove data from the interface, and control global display options. The functions are briefly discussed below.



Options for adding required data

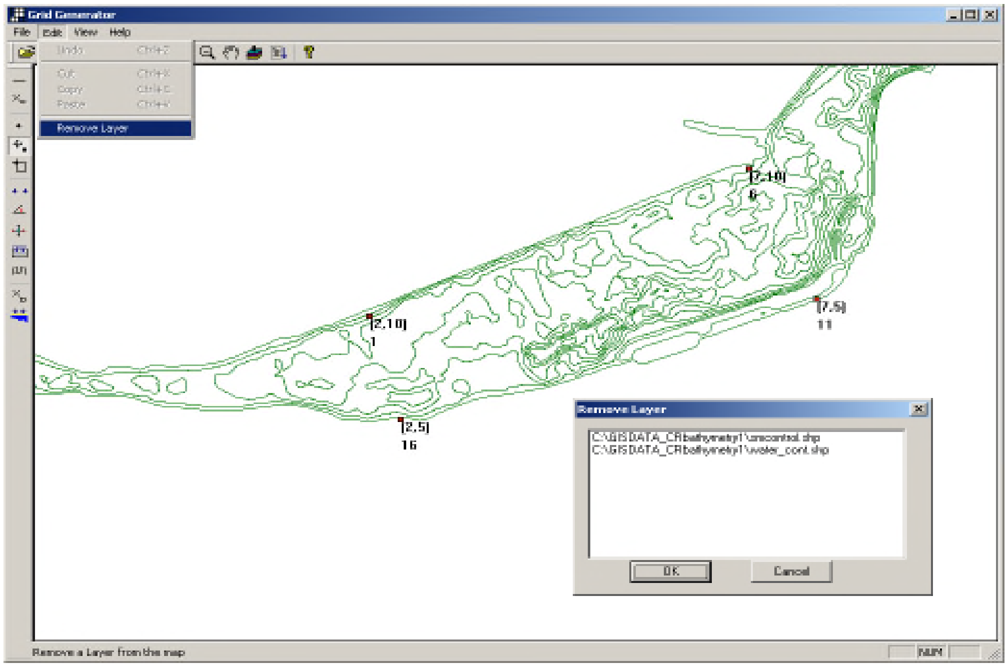
Pulldown for exporting finished grid

Figure 3. Import/Export Options under the File Pulldown.

The File pulldown menu (Figure 3) contains functions that allow the user to import and export existing data. Selecting the *Add Water Boundary* command will enable the user to select an existing shoreline shape file and load it into the VOGG project. The *Import Control Point Layer* command will allow the user to select an existing control point shape file and load it into the project.

The *Add Grid Layer* command allows the user to select a grid layer to be imported as a shape file or matlab text file. The *Import Waterbody UTM File* command will allow the user to select an existing shoreline UTM file and load it into the project. The *Export Grid Data* command allows the user to export grid data developed by VOGG to a selected folder.

Once data has been imported to the VOGG Grid Generator, any changes made to the data will be permanent. However, the layers in VOGG can not be saved as a project like ArcView. If the project is closed, all data must be imported again. To remove data, the Edit-> *Remove Layer* command will allow the user to do so (Figure 4). Global display commands can also be accessed through the pulldown menus (Figure 5).



C:\GISDATA_CR\bathymetry1\unscontrol.shp
C:\GISDATA_CR\bathymetry1\water_cont.shp

Figure 4. The Remove Layer Option under the Edit pulldown menu.

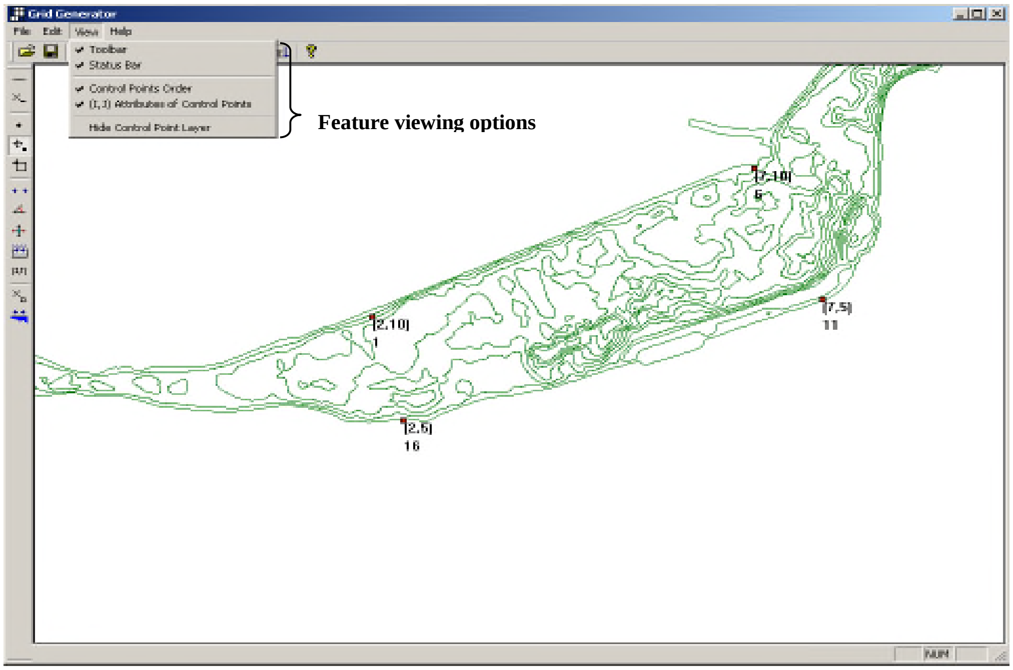


Figure 5. Global display command options under the View pulldown menu.

6. User Interface Tutorial

This exercise will go through the necessary steps in developing a curvilinear orthogonal grid using the VOGG Grid Generator Interface.

Before developing a grid for a particular study area using the VOGG Grid Generator, digital data representing a shoreline boundary must be obtained. These data can be in ESRI Shapefile or Waterbody UTM format. Many sources for these data exist. The first step the tutorial will take will be to obtain shoreline data in shapefile format from the NOAA Coastal Services Center website.

1. Using a web browser, navigate to: http://www.csc.noaa.gov/opis/html/datadown.htm

This site contains a variety of coastal data, but at this stage we are only interested in shoreline data. Scroll down to the Physical Resources section of the page and find the Regional Shoreline data layer. Select the zipped shapefile sshr70k.zip, and save the file to C:\VOGG\Tutorial Data\. Unzip the files and save them in the same directory.

2. Navigate to the Executables\ folder where GRID.exe is located.

3. Double clicking on GRID.exe will open the interface of Grid Generator window (Figure 1).

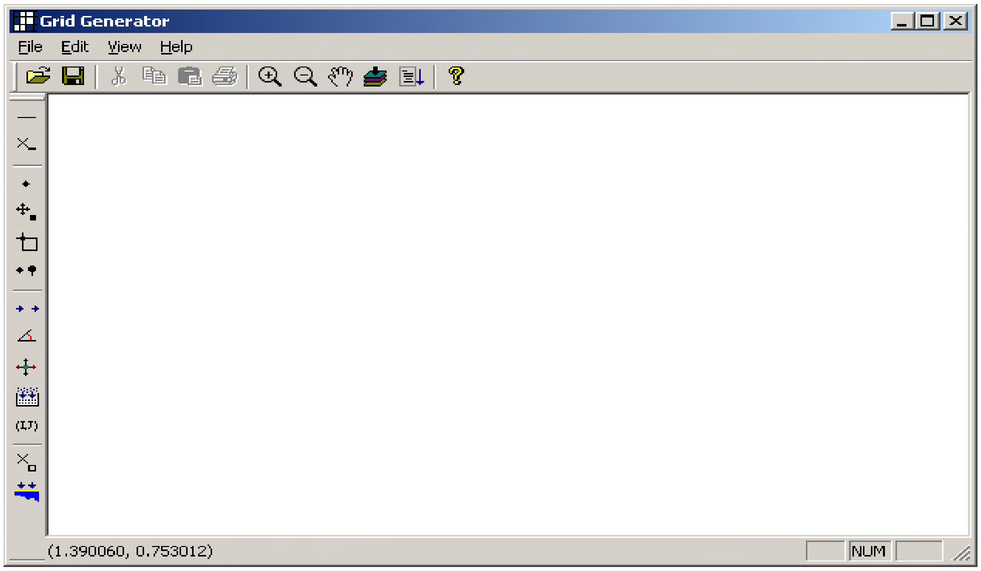
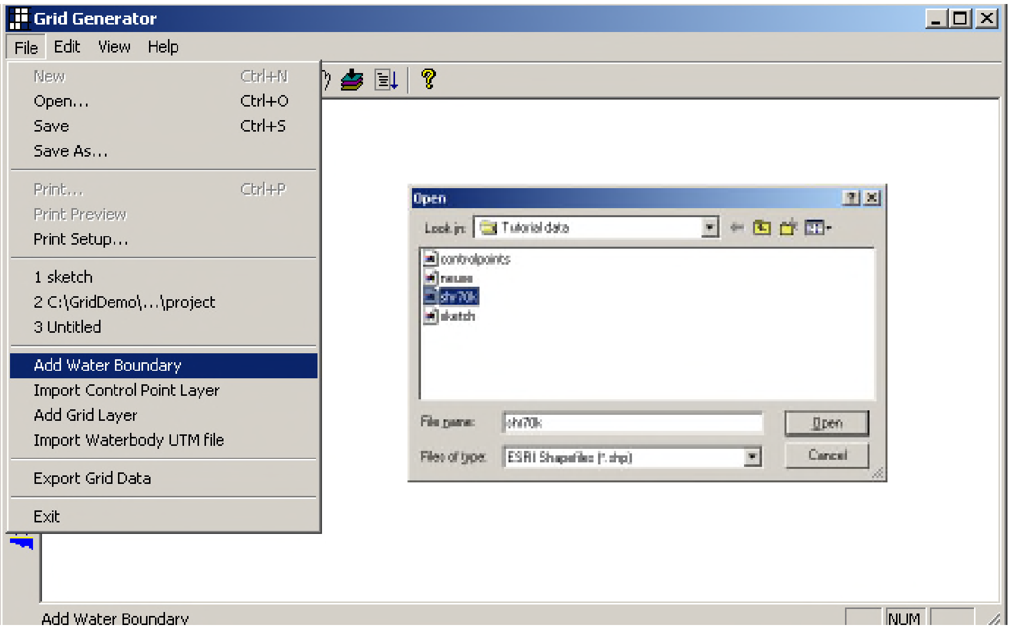


Figure 1. Grid Generator Window.

4. Select the File pulldown on the menu bar and select *Add Water Boundary* (Figure 2). Select sshr70k.shp in the /*Tutorial Data* folder and click OK. The Grid Generator screen should look similar to Figure 3.
   Note: Keep in mind your choice of data projection: If you wish to use the resulting grid data as a layer in an existing GIS, have that layer initially projected in that project's projection.



Look in: Tutorial data

File name: shr70k

Files of type: ESRI Shapefiles (*.shp)

Figure 2. Adding a Water Boundary.

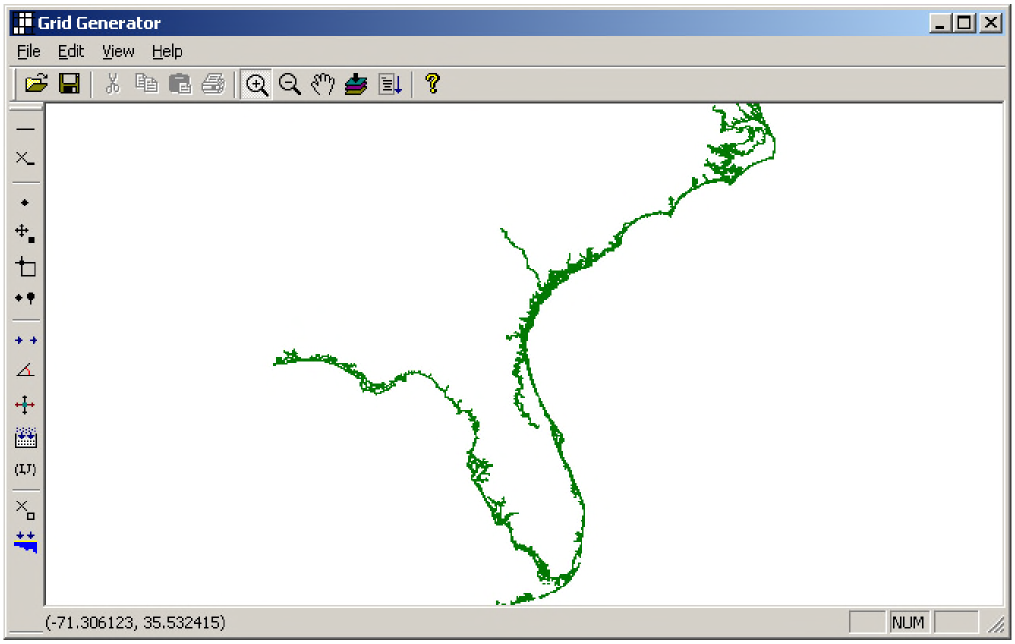
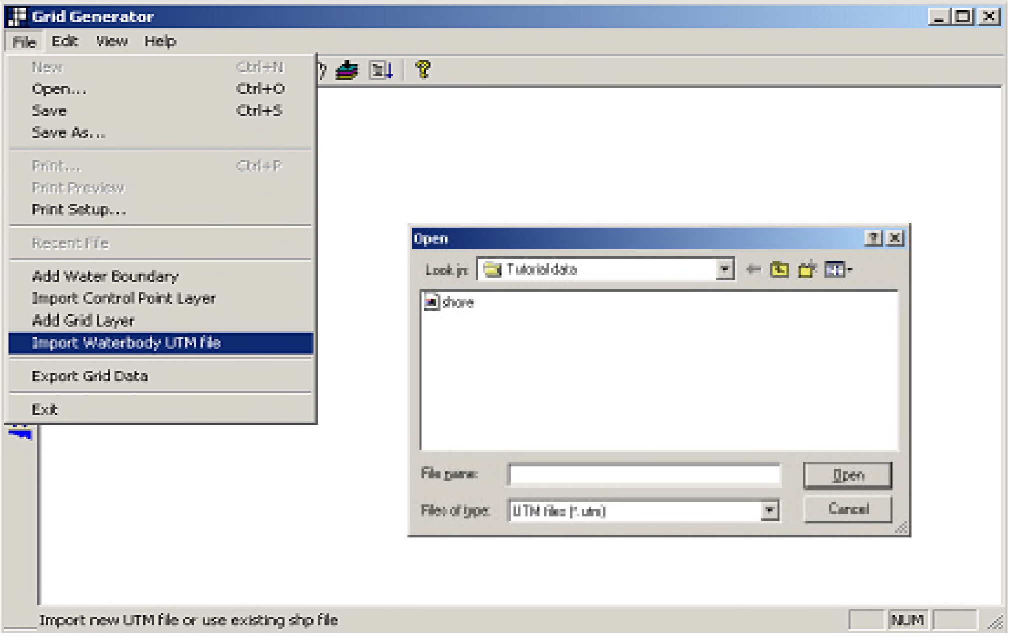


Figure 3. The Water Boundary theme in the Grid Generator interface.

The theme we have imported to the Grid Generator contains shoreline data for the entire southeast coast of the United States. This scale may be too large for our use, so this is a good opportunity to learn how to import a shoreline file of another format.

5. In order to remove a shapefile to begin another project, remove the sshr70k.shp shoreline file by selecting the Edit-Remove Layer pulldown option, and selecting the sshr70k.shp file in the Remove Layer dialog box. Now we can import a new shoreline file to work with.

6. Select the File pulldown on the menu bar and select *Import Waterbody UTM file*. Select shore.utm in the /*Tutorial Data* folder (Figures 4&5).



Look in: Tutorial data

shore

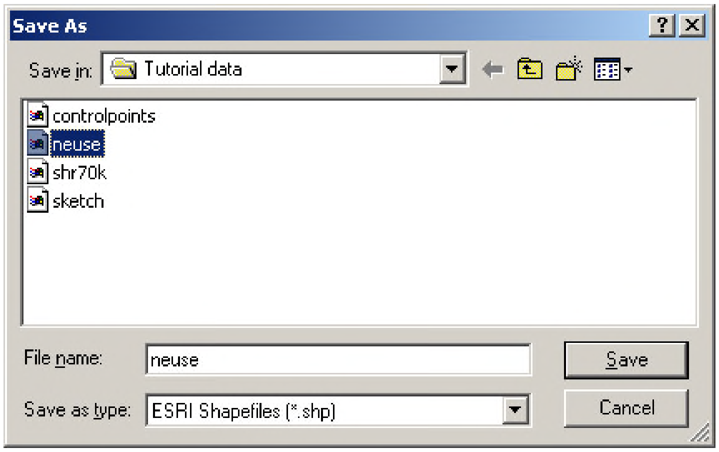
File name:

Files of type: UTM files (*.utm)

Import new UTM file or use existing shp file

Figure 4. Selecting *Import Waterbody UTM file.*

7. After selecting the UTM file, you will be prompted to save the file as a shape file. Save the file as /*Tutorial Data*/*Neuse* in the SAVE AS dialog box (Figure 5). Select SAVE, and neuse.shp will be imported into the interface (Figure 6). In the future, the *neuse.shp* shapefile can be added using the *Add Water Boundary Tool*.



Save in: Tutorial data

controlpoints
neuse
shr70k
sketch

File name: neuse

Save as type: ESRI Shapefiles (*.shp)

Figure 5. Save data in shape file format in SAVE AS window.

8. Select the zoom tool on the Top Toolbar and zoom to the boxed area shown in Figure 6.

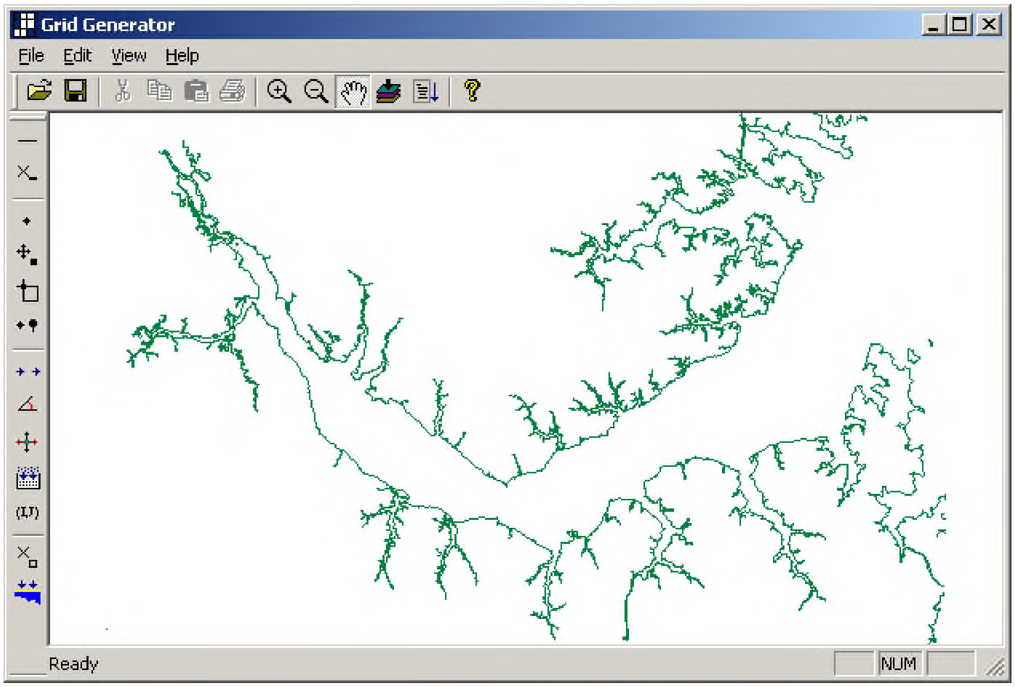


Figure 6. The Neuse waterbody shape file, and the area to be zoomed in.

9. Select the *Add Sketch Grid Lines Tool*. This tool will assist in visualizing the model domain and in assigning the placement of corner points.

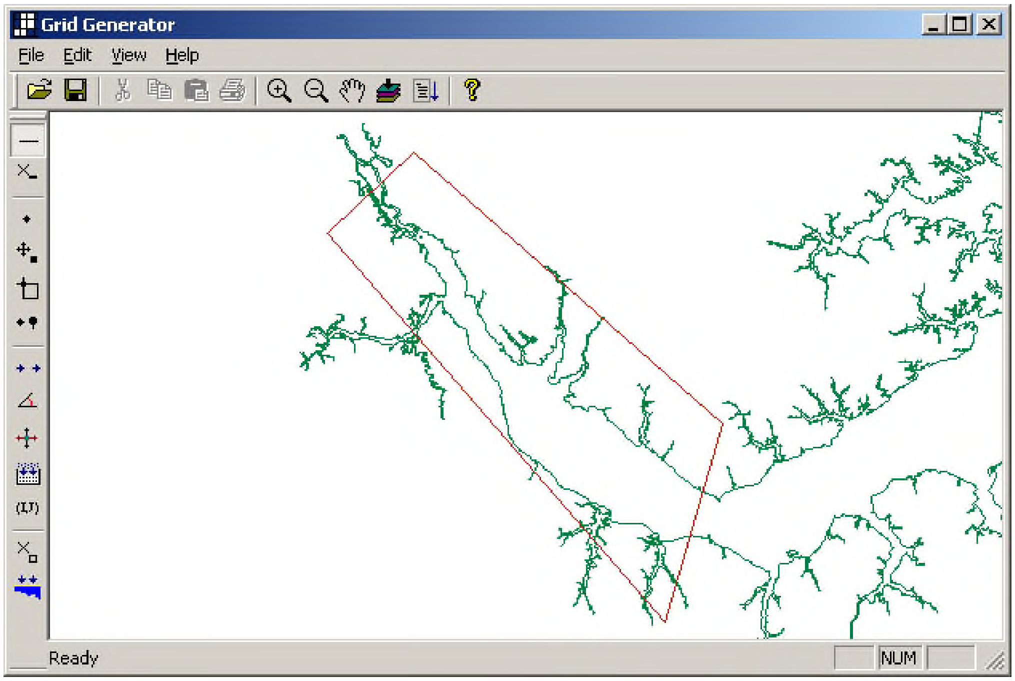


Figure 7. The sketched boundary of the model domain.

10. Click the left mouse button to start sketching the approximate boundary of the model domain. Sketch a box that looks approximately like that shown in Figure 7 and double-click the right mouse button to end the sketch. You will be prompted to name the shapefile. Name it sketch.shp.

11. Use the *Add Control Points Tool* to add points at a specified location (Figure 8). Creating the first control point by clicking on the specified location in Figure 8 will prompt you to save the points you create as a shape file. Save the file as *controlpoint.shp*.

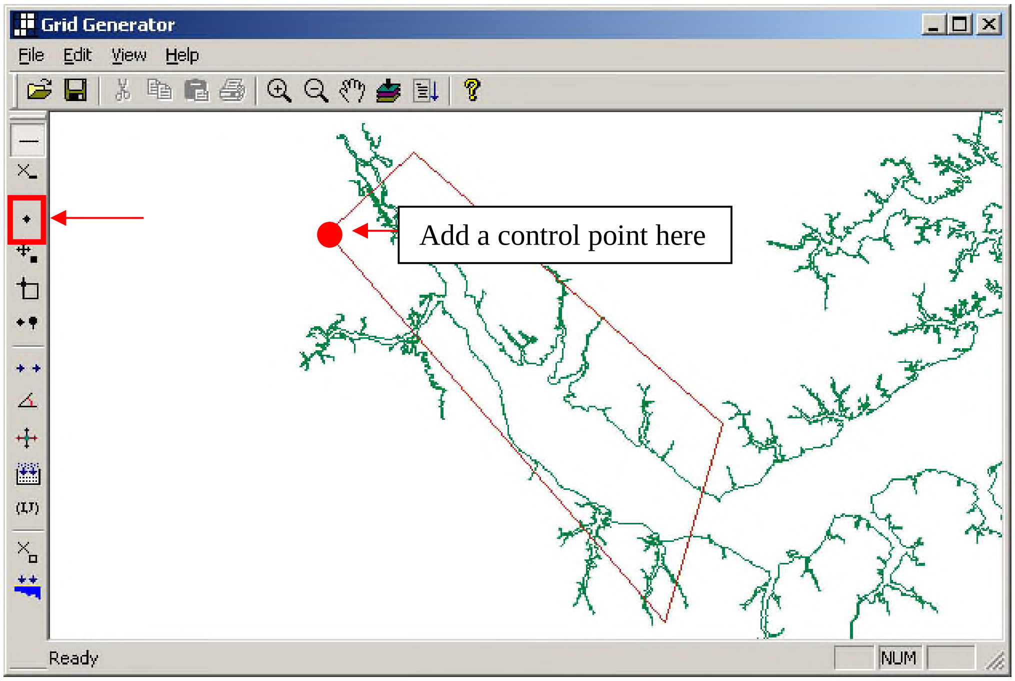


Figure 8. ***Add Control Points Tool.***

12. Continue to add an additional four control points as shown in Figure 9.

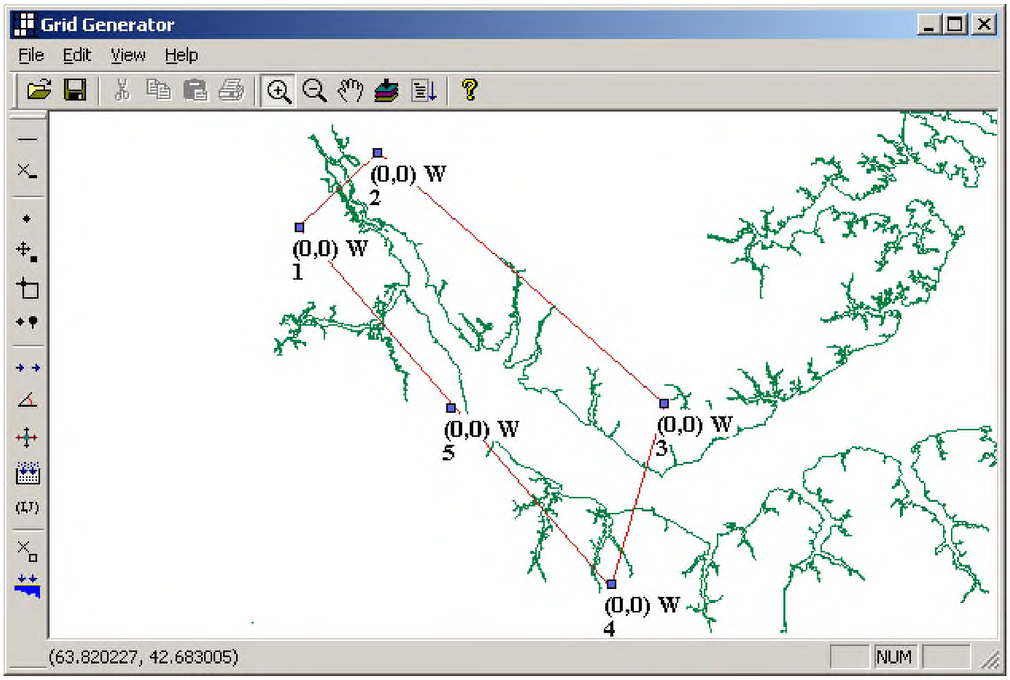


Figure 9. **Adding control points**

13. Select the *Select Control Points Tool*. Select the node shown in Figure 10 by drawing a box around the node with your mouse. Once the node has been selected, press the *Delete* key on your keyboard to remove the node. When asked if you want to delete the selected control point, click Yes.

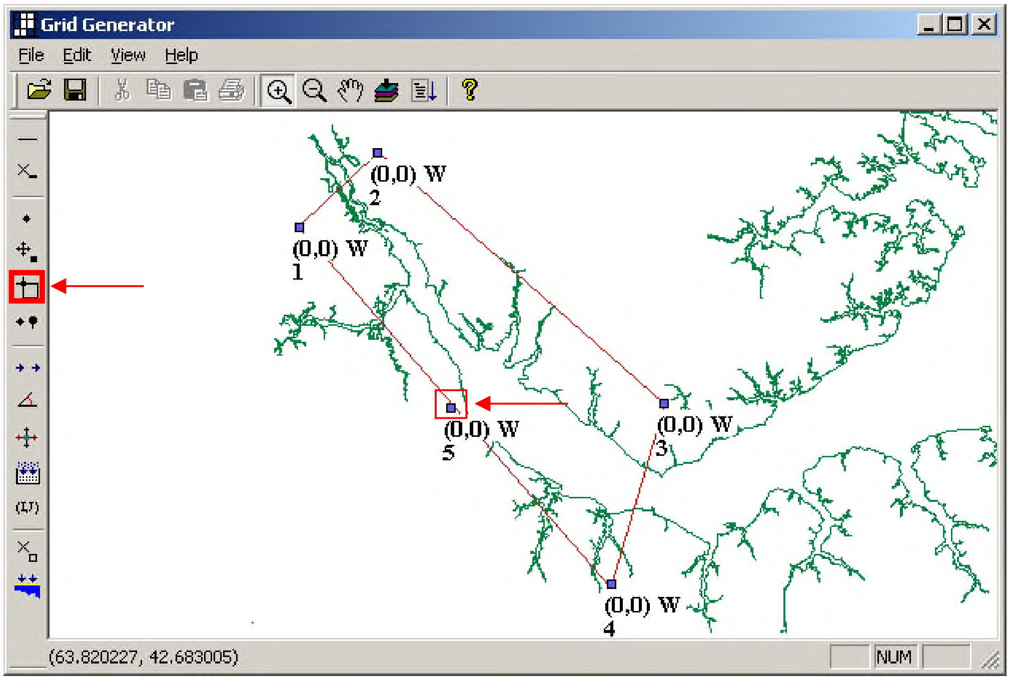


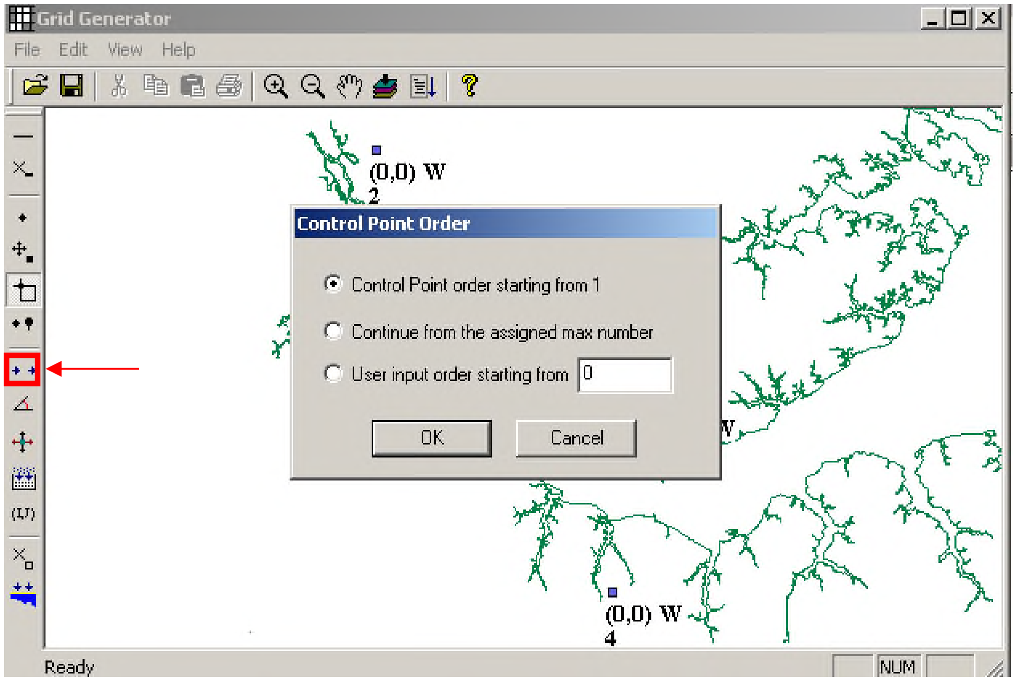
Figure 10. Deleting control points.

14. Remove the sketch line theme from the display by selecting the *Edit /Remove Layer* pulldown function and selecting the *sketch.shp* file for removal.

15. Select the *Generate Control Point Order Tool*, and select the *Control Point Order Starting From 1* radio button to assign the first control point number (Figure 11).

16. Select the first control point that you plotted to assign a value of 1. The control point will turn red when it has been selected, and the bottom number identifying the order number will change to 1. Continue to assign the order to each of the control points in clockwise direction as shown in Figure 12.

* *Note: Any corner point can be selected as the first point. However, the order of points must be assigned in clockwise fashion in this example.*



Control Point Order

● Control Point order starting from 1

○ Continue from the assigned max number

○ User input order starting from 0

Figure 11. Control Point Order dialog box.

17. Now that the order of control points has been assigned, the alignment of the boundary needs to be indicated. Using the *Specify Boundary Direction Tool* (Figure 13), select control point number 1. The boundary segment in the clockwise direction from this control point (the boundary between point 1 and point 2) is facing west, so click the radio button to select West in the *Direction Settings* dialog box.

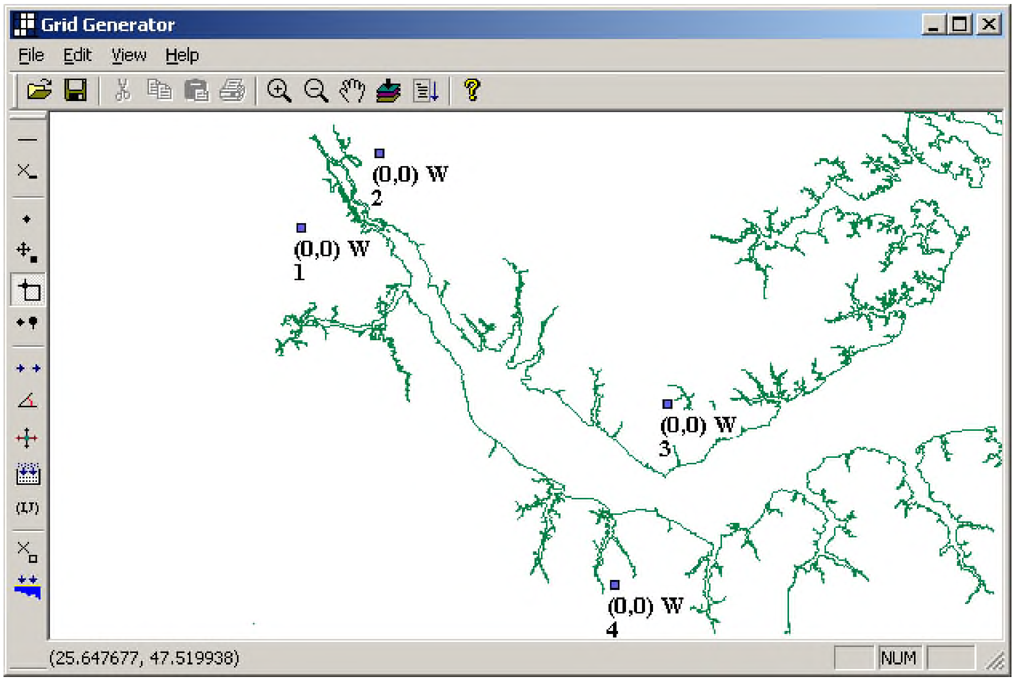
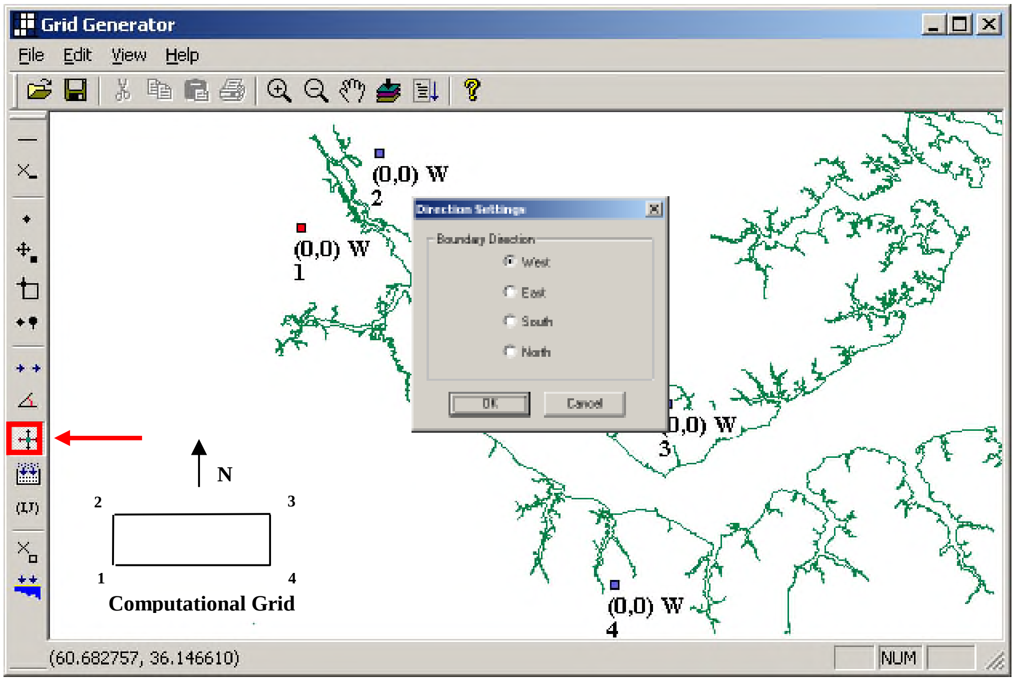


Figure 12. Control Points with assigned order.



Direction Settings

Boundary Direction

● West
○ East
○ South
○ North

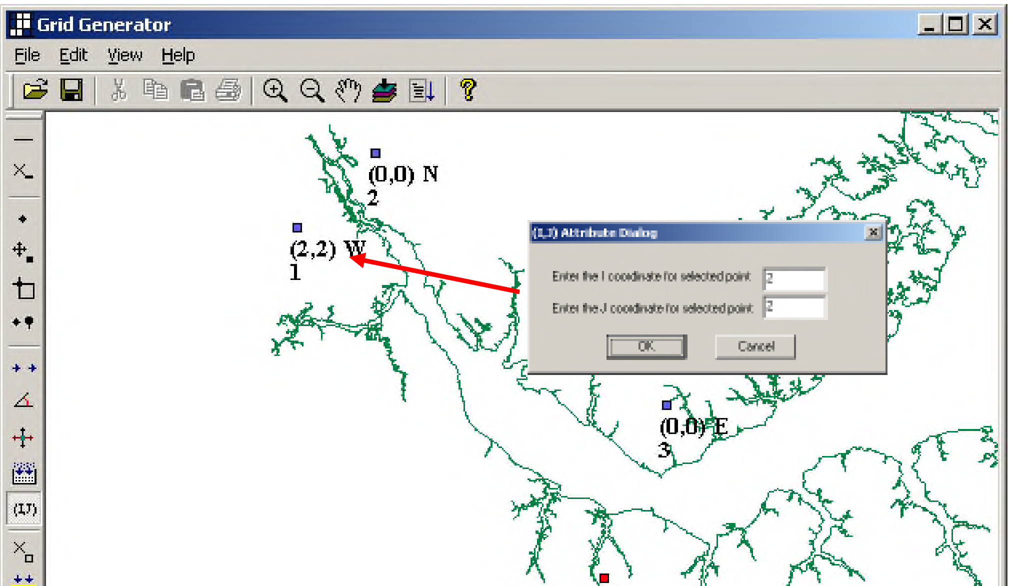
OK Cancel

Figure 13. Assigning Control Point Boundary Direction.

18. Continue assigning the directions for each control point. Starting with node 1, the assignments should be: 1=West, 2=North, 3=East, 4=South.

We will now assign the I,J locations of the control points manually. Later, we will run through the procedure of automatically assigning the locations.

19. Turn on the *Manual I,J Assignment Tool* (Figure 14). Select control point 1 and assign an I coordinate of 2, and a J coordinate of 2 in the I,J Attribute dialog box. Continue assigning I,J coordinates to the rest of the control points, using the following values: 2=2,12, 3=42,12, 4=42,2. The Grid Generator can be run at this point, but instead we will use the *Automatic I,J Assignment Tool*.



(I,J) Attribute Dialog

Enter the I coordinate for selected point: 2

Enter the J coordinate for selected point: 2

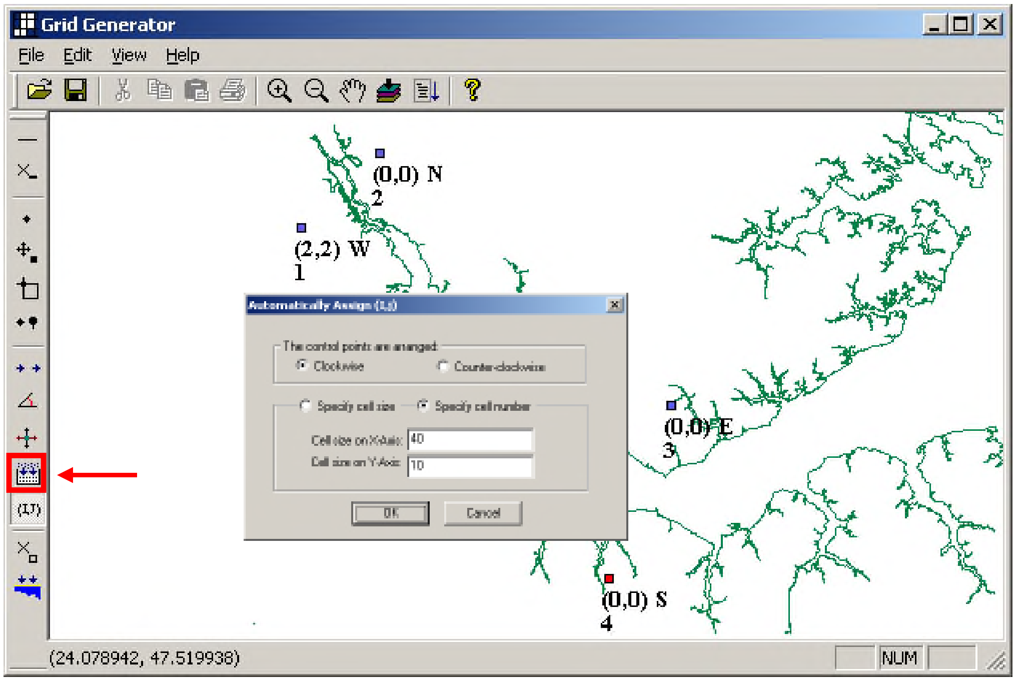
OK Cancel

Figure 14. Manual assignment of I,J coordinates.

20. To assign I, J coordinates automatically, select the *Generate (I,J) Automatically Tool*. The Automatically Assign (I,J) dialog box will appear (Figure 15).

21. Select the Clockwise option in the "The control points are arranged" section if it is not selected already, to indicate that you have assigned the control points in that fashion. Also select the Specify Cell Number radio button and type in a value of 40 for the X-axis and 10 for the Y-axis. Select OK, and then the Automatically Generate (I,J) dialog box will appear. Select the Calculate button, and then Exit. The control points are now numbered with I,J coordinates.

Note: The configuration of the resulting coordinates may differ from the parameters you specified in the (I,J) Attribute Dialog Box, especially as the configuration strays farther from a rectangular feature. The resulting grid represents the closest possible representation to the desired grid dimensions.



Automatically Assign (I,J)

The control points are arranged
● Clockwise ○ Counter-clockwise

○ Specify cell size ● Specify cell number

Cell size on X-Axis: 40
Cell size on Y-Axis: 10

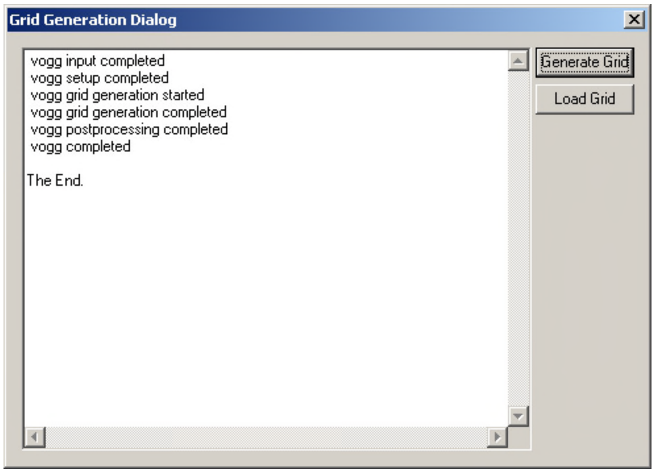
OK Cancel

(24.078942, 47.519938)

Figure 15. Automatically Assign I,J dialog box.

22. Alternatively, a cell size can be assigned in the dialog box. Select the *Generate (I,J) Automatically Tool* and select the Specify Cell Size radio button in the dialog box. The cell size should be assigned based on the map units the spatial data is in. In this case, the data is in kilometers, so assign an X-axis cell size of 1, and a Y-axis cell size of 1. Select OK.

23. After all of the control points have been assigned I, J coordinates, run the VOGG Grid Generator using the *Generate Curvilinear Grid Tool* located on the top tool bar*:*



```
vogg input completed
vogg setup completed
vogg grid generation started
vogg grid generation completed
vogg postprocessing completed
vogg completed

The End.
```

Figure 16. Grid Generation dialog box.

Selecting the tool will initiate the Grid Generation dialog box if a grid does not currently exist in the VOGG project. Select *Generate Grid* (Figure 16).

Grid Generation may take a few minutes. The grid layout for this exercise was chosen for the short processing time. In a real application, the user may wish to reconfigure the control points to establish a more suitable grid, but the processing step may take much longer.

When the Grid Generator has completed the run, "The End" will appear in the Grid Generation dialog box (Figure 16). Select Load Grid and name the grid "Grid.shp." The Grid that is produced should look similar to that in Figure 17. To fine tune the grid to the study area, the user can manually adjust the locations of control points using the *Move Control Points Tool*, or adjust the theta values using the *Assign Control Point Theta Tool*.

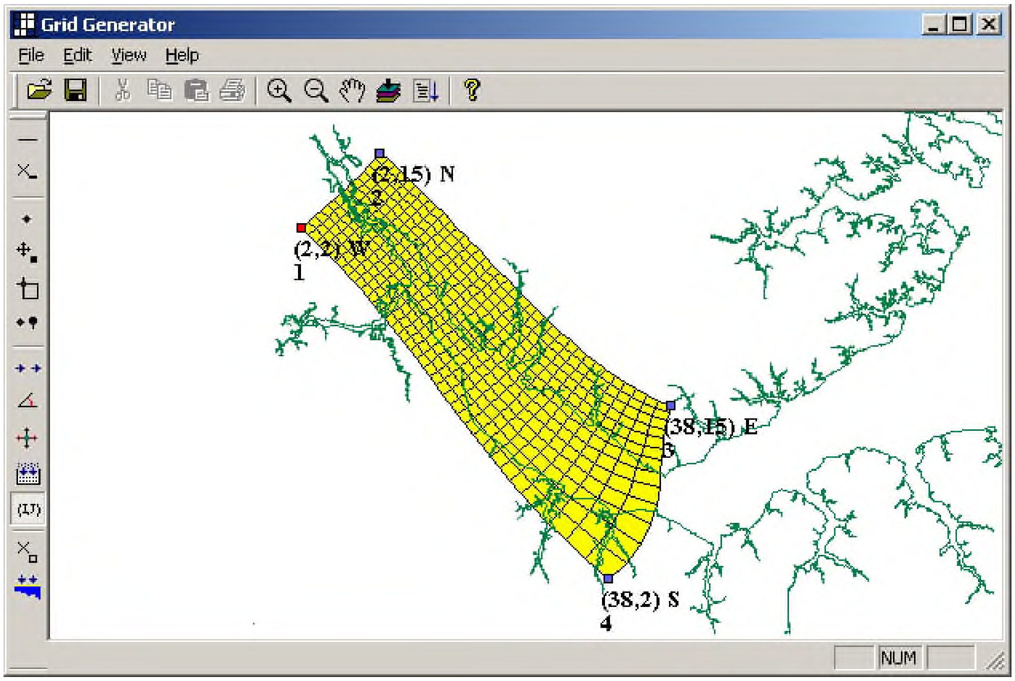


Figure 17. Grid produced by the VOGG Grid Generator.

24. Select the Move Control Point Tool from the Toolbar. Move the cursor over control point 1, hold the left mouse button down, and drag the control point to the location shown in Figure 18. To produce a new grid, it's a good idea to reassign the I,J coordinates if they were assigned based on a cell size. Go back to Step 16 to reassign I,J values based on cell size. Select *Generate Grid* after new I,J values have been assigned.

The new grid may still need to be adjusted. The next steps will describe another method of modifying the grid to better represent the boundaries of the study area.

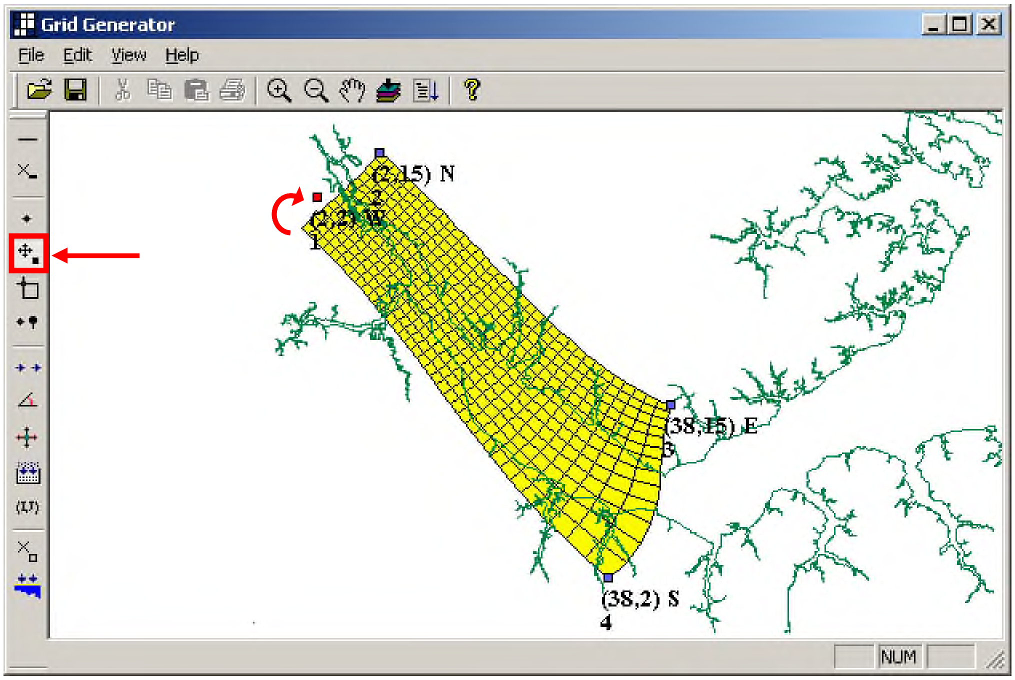
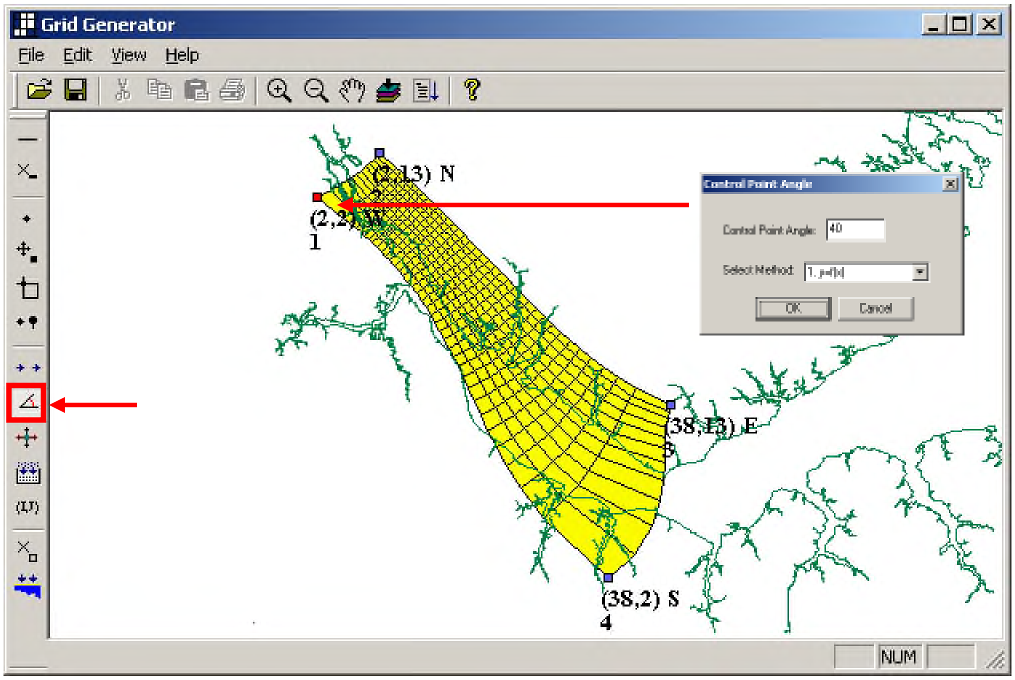


Figure 18. Moving Control Point.

25. Select the *Assign Control Point Theta Too1* and select control point 1 (Figure 19). Assign a value of 40 to control point 1 and keep the current method. Select OK.



Control Point Angle: 40

Select Method: 1. j=f(x)

Figure 19. Assigning control point theta values.

Continue the assignment of theta values to each of the control points using the following values: 2 = -60, 3 = -95, 4 = 170. Also, for control point 4, change the method to #2; y=f(x).

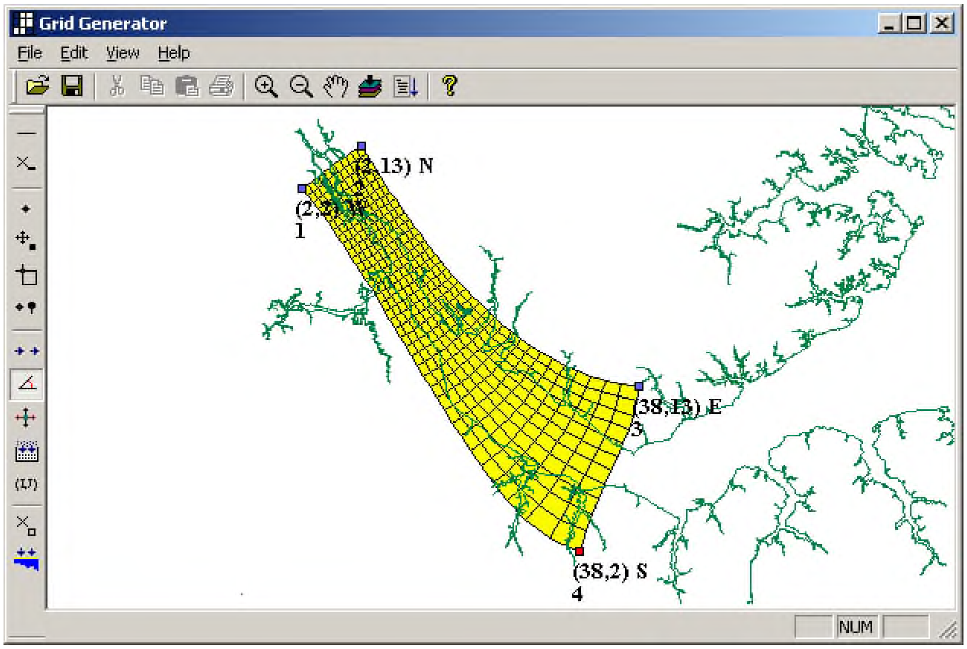


Figure 20. Grid produced after adjusting control point theta values.

26. Rerun VOGG using the *Generate Curvilinear Grid Tool*. The resulting grid should look different and (hopefully) be a better fit to the Neuse River shoreline (Figure 20).

27. Cells can be removed from the grid to mimic the Neuse River shoreline more accurately. Select cells that fall over land using the *Delete Cell Tool*, similarly to Figure 21. The mouse can be used to draw a box surrounding a group of cells you wish to remove. The shift key can also be used to select multiple cells or groups of cells at a time. To delete the cells from the grid, press the Delete key on your keyboard. This will permanently delete these features (Figure 22). We can now assign bathymetry values to the remaining cells.

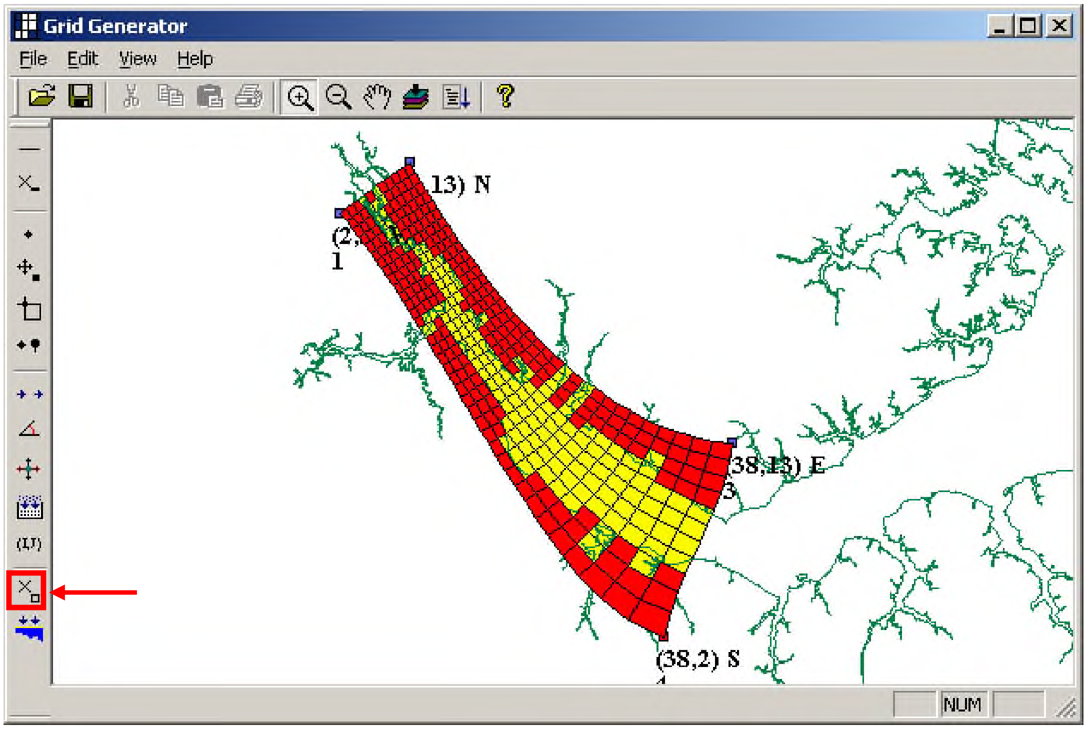


Figure 21. Cells to be deleted.

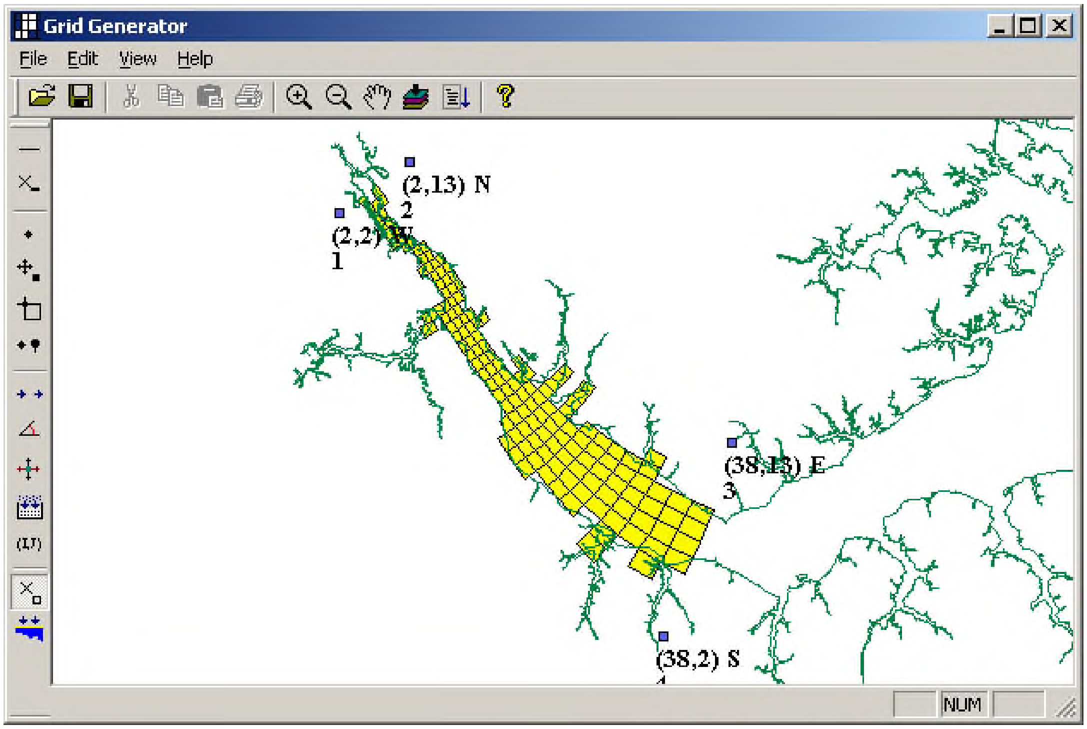
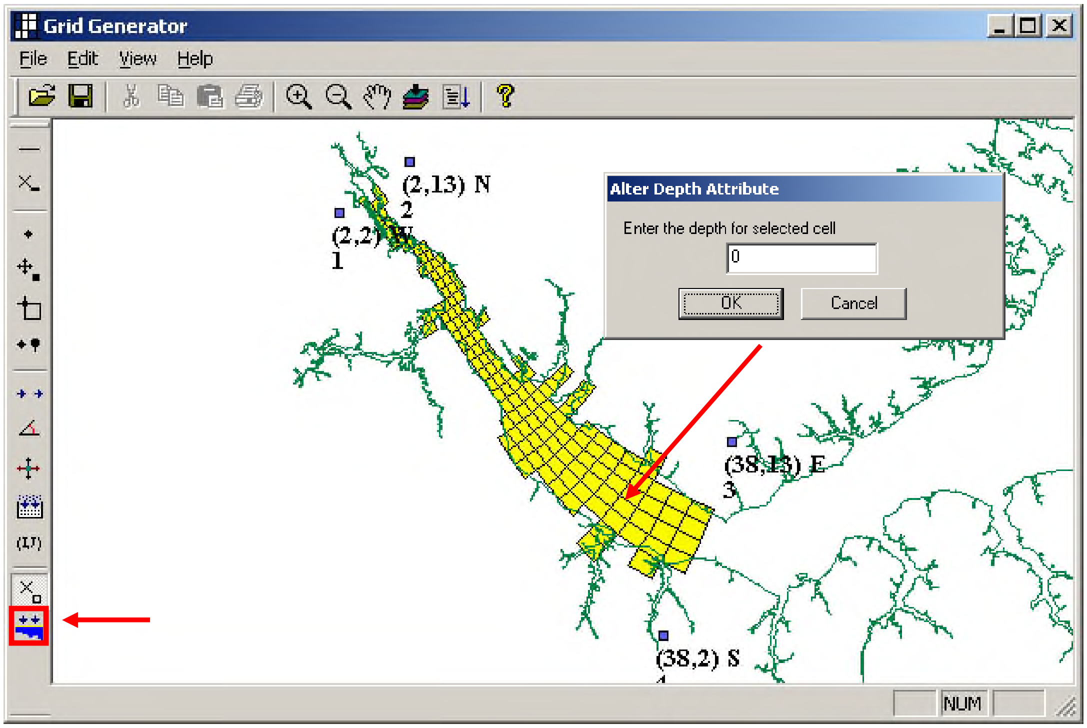


Figure 22. The new grid.

28. Select the *Edit Water Depth Tool* shown in Figure 23 and select one of the downstream cells in the grid. The Alter Depth Attribute should be displayed. The native map data is in kilometers, so to assign a depth of 5 meters, enter a value of 0.005 in the Alter Depth Attribute dialog box. Select OK. The depth of this cell has been recorded in the *.dbf file of the grid shapefile.



Alter Depth Attribute

Enter the depth for selected cell

0

Figure 23. Assigning depth attributes to the remaining grid cells.

29. Select the cell you have modified and record the I,J coordinates shown in the lower left corner of the interface. Navigate to Executables\dxdy.inp and open the file with a spreadsheet or text editor. The first two columns of the dxdy.inp file show the I and J indices for all cells in the mesh. Find the I,J indices for the cell you modified and make sure the changes have been recorded.

7. References

Fletcher, C. A. J., 1991: Computational Techniques for Fluid Flow, Volume II. Springer-Verlag, Berlin.

Hamrick, J. M., 1996: Users manual for the environmental fluid dynamic computer code. The College of William and Mary, Virginia Institute of Marine Science, Special Report 328.

Knupp, P., and S. Steinberg, 1993: Fundamentals of Grid Generation. CRC Press, Boca Raton.

Mobley, C. D., and R. J. Stewart, 1980: On numerical generation of boundary fitted orthogonal curvilinear coordinate systems. . *Journal of Computational Physics*, **34**, 124-135.

Ryskin, G., and L. G. Leal, 1983: Orthogonal mapping. *Journal of Computational Physics*, **50**, 71-100.

Thompson, J. F., Z. U. A. Warsi, and C. W. Mastin, 1985: Numerical Grid Generation, Foundations and Applications. North-Holland, New York.

Appendix A: Grid Generation Theory

A.1 Generation of Orthogonal Grids in the Computational Domain

For two-dimensional orthogonal grid generation in the computational (ξ,η) domain, the physical grid (x,y) coordinates satisfy the elliptic equations

$$\frac{\partial}{\partial\xi}\left(\frac{h_\eta}{h_\xi}\frac{\partial x}{\partial\xi}\right)+\frac{\partial}{\partial\eta}\left(\frac{h_\xi}{h_\eta}\frac{\partial x}{\partial\eta}\right)=0 \qquad (1)$$

$$\frac{\partial}{\partial\xi}\left(\frac{h_\eta}{h_\xi}\frac{\partial y}{\partial\xi}\right)+\frac{\partial}{\partial\eta}\left(\frac{h_\xi}{h_\eta}\frac{\partial y}{\partial\eta}\right)=0 \qquad (2)$$

(Knupp and Steinberg, 1993; Thompson, Warsi and Mastin, 1985) where the scale factors are given by

$$h_\xi=\sqrt{\left(\frac{\partial x}{\partial\xi}\right)^2+\left(\frac{\partial y}{\partial\xi}\right)^2} \qquad (3)$$

$$h_\eta=\sqrt{\left(\frac{\partial x}{\partial\eta}\right)^2+\left(\frac{\partial y}{\partial\eta}\right)^2} \qquad (4)$$

Defining the scale factor ratio

$$f=\frac{h_\eta}{h_\xi} \qquad (5)$$

Equations (1) and (2) can be written as

$$\frac{\partial}{\partial\xi}\left(f\frac{\partial x}{\partial\xi}\right)+\frac{\partial}{\partial\eta}\left(\frac{1}{f}\frac{\partial x}{\partial\eta}\right)=0 \qquad (6)$$

$$\frac{\partial}{\partial\xi}\left(f\frac{\partial y}{\partial\xi}\right)+\frac{\partial}{\partial\eta}\left(\frac{1}{f}\frac{\partial y}{\partial\eta}\right)=0 \qquad (7)$$

For conformal orthogonal grids, the scale factor is unity, while a constant value of *f* corresponds to a quasi-conformal orthogonal grid. Methods for generation of conformal and quasi-conformal grids are quite restrictive and are often limited to rectangular computational domains (Mobley and Stewart, 1980). Allowing *f* to vary throughout the domain allows the computational domain to be block rectangular. The variable *f* methods presented by Ryskin and Leal (1983) are representative, and a description of their weak constraint method implemented in the GEFDC grid generation code (Hamrick, 1996) follows.

Solution of equations (6) and (7) for the physical coordinates (*x*,*y*) at discrete computational grid points proceeds by first constructing a computational grid domain corresponding the physical grid domain. Figure 1 shows the boundary of a water body in the physical domain, while Figure 2 shows a corresponding approximation of the water body boundary by a curvilinear block rectangular region in the physical domain. Note that the vertices of the multi-block rectangular region in Figure 2, shown as bold circles, define right angle corners. Figure 3 shows a computational domain corresponding the physical domain in Figure 2. The computation domain has a block rectangular shape corresponding the curvilinear linear block rectangular shape of the physical domain in Figure 2.

A number of approaches can be used for specifying boundary conditions for the solution of (6) and (7) in the computational domain. In the weak constraint method (Ryskin and Leal, 1983), the boundary correspondence is defined by specifying the physical coordinates (*x*,*y*) at each vertex on the computation grid boundary. The weak constraint method also requires specification of the scale factor ratio at all vertices in the computational domain. This can be done by calculating (3), (4), and (5) at boundary vertices, followed by interpolation of *f* to the interior vertices of the computational grid, which results in (6) and (7) becoming nonlinearly coupled. The iterative algorithm for the coupled equations is

$$\frac{\partial}{\partial \xi}\left(f^{(n-1)} \frac{\partial x^{(n)}}{\partial \xi} \right) + \frac{\partial}{\partial \eta}\left(\frac{1}{f^{(n-1)}} \frac{\partial x^{(n)}}{\partial \eta} \right) = 0 \tag{8}$$

$$\frac{\partial}{\partial \xi}\left(f^{(n-1)} \frac{\partial y^{(n)}}{\partial \xi} \right) + \frac{\partial}{\partial \eta}\left(\frac{1}{f^{(n-1)}} \frac{\partial y^{(n)}}{\partial \eta} \right) = 0 \tag{9}$$

where *n* denotes the current iteration level. Note that (8) and (9) are linear at a given iteration level. The scale factor *f* is calculated at vertices along the computational domain boundary and interpolated to the interior. A smooth distribution of *f* in the interior results by using the solution of

$$\frac{\partial}{\partial \xi}\left(\frac{\partial f^{(n-1)}}{\partial \xi} \right) + \frac{\partial}{\partial \eta}\left(\frac{\partial f^{(n-1)}}{\partial \eta} \right) = 0 \tag{10}$$

as the interpolator. To begin the iteration, *f(0)* is determined by a initial grid generated by the solution of

$$\frac{\partial}{\partial \xi}\left(\frac{\partial x^{(0)}}{\partial \xi}\right)+\frac{\partial}{\partial \eta}\left(\frac{\partial x^{(0)}}{\partial \eta}\right)=0 \tag{11}$$

$$\frac{\partial}{\partial \xi}\left(\frac{\partial y^{(0)}}{\partial \xi}\right)+\frac{\partial}{\partial \eta}\left(\frac{\partial y^{(0)}}{\partial \eta}\right)=0 \tag{12}$$

All of the generation equations, (8) through (12), represent elliptic boundary value problems and are readily discretized by vertex centered finite differences in the computation domain. The resulting systems of linear equations for the interior vertex located unknowns are symmetric, positive definite and readily solved by relaxation or conjugate gradient schemes.

The quality of the grid obtained by the solution of (8-10) is fairly sensitive to the assignment of (x,y) coordinates to boundary vertices of the computational grid. Figure 4 shows the initial grid in the physical domain, corresponding to the physical region and computational grid in Figures 2 and 3, determined by the solution of (11) and (12) with approximately equal spacing of cell vertices along each boundary segment. The final grid obtained by the solution of (8), (9), and (10) is shown in Figure 5. The quality of the grid in Figure 5 is rather poor in the vicinity of x = 4.5 and y = 3. The quality of the grid can be significantly improved by redistribution of the grid vertices along the physical region boundary. Figure 6 shown an improved grid achieve by a variable spacing of the grid vertices along the boundary. Selection of the grid vertex distribution along the physical domain boundary, which corresponds to specification of (x,y) at the boundary vertices of the computational grid can be viewed as an iterative process guide by intuition and experience.

A.2 Generation of Orthogonal Grids in the Physical Domain

Orthogonal grids can also be generated in the physical plane. The computational grid coordinates can be show to satisfy

$$\frac{\partial}{\partial x}\left(\frac{h_\xi}{h_\eta}\frac{\partial \xi}{\partial x}\right)+\frac{\partial}{\partial y}\left(\frac{h_\xi}{h_\eta}\frac{\partial \xi}{\partial y}\right)=0 \tag{13}$$

$$\frac{\partial}{\partial x}\left(\frac{h_\eta}{h_\xi}\frac{\partial \eta}{\partial x}\right)+\frac{\partial}{\partial y}\left(\frac{h_\eta}{h_\xi}\frac{\partial \eta}{\partial y}\right)=0 \tag{14}$$

in the interior of the physical domain, and

$$h_\xi\frac{\partial \xi}{\partial x}=h_\eta\frac{\partial \eta}{\partial y} \tag{15}$$

$$h_\xi \frac{\partial \xi}{\partial y} = -h_\eta \frac{\partial \eta}{\partial x} \qquad (16)$$

on the boundaries (Fletcher, 1991). Equations (15) and (16) also imply

$$\frac{\partial \xi}{\partial x}\frac{\partial \eta}{\partial x} + \frac{\partial \xi}{\partial y}\frac{\partial \eta}{\partial y} = 0 \qquad (17)$$

or

$$\frac{\partial x}{\partial \xi}\frac{\partial x}{\partial \eta} + \frac{\partial y}{\partial \xi}\frac{\partial y}{\partial \eta} = 0 \qquad (18)$$

To solve the grid generation problem in the physical domain, boundary conditions satisfying (17) must be provided. Noting that boundary segments in the physical domain correspond to lines of constant ξ and η, the boundary can be divided into two types of segments. On the boundary segments denoted by Γ_ξ, the values of ξ are specified as constant and on the boundary segments denoted by Γ_η, the values of η are specified as constant. Thus the specified values of ξ and η on the segments Γ_ξ and Γ_η, provide boundary conditions for (13) and (14), respectively.

What remains is the specification of boundary conditions for (13) on Γ_η boundary segments and for (14) on Γ_ξ boundary segments. On the Γ_ξ boundaries the unit vectors tangential and normal to the boundary are defined by

$$\begin{aligned} \vec{u}_t &= u_{tx}\vec{i} + u_{ty}\vec{j} \\ \vec{u}_n &= -u_{ty}\vec{i} + u_{tx}\vec{j} \end{aligned} \qquad (19)$$

and since the gradient of ξ along theses boundary segments is zero

$$G_{t\xi} = \vec{u}_t \bullet \nabla \xi = u_{tx}\frac{\partial \xi}{\partial x} + u_{ty}\frac{\partial \xi}{\partial y} = 0 \quad : \quad \Gamma_\xi \qquad (20)$$

holds. On the Γ_η boundaries the unit vectors tangential and normal to the boundary are defined by

$$\begin{aligned} \vec{v}_t &= v_{tx}\vec{i} + v_{ty}\vec{j} \\ \vec{v}_n &= -v_{ty}\vec{i} + v_{tx}\vec{j} \end{aligned} \qquad (21)$$

and since the gradient of η along theses boundary segments is zero

$$G_{t\eta} = \vec{v}_t \bullet \nabla \eta = v_{tx} \frac{\partial \eta}{\partial x} + v_{ty} \frac{\partial \eta}{\partial y} = 0 \quad : \quad \Gamma_\eta \tag{22}$$

holds. On the Γη boundary segments, the boundary condition of ξ is proposed to be

$$G_{n\xi} = \vec{v}_n \bullet \left(\frac{h_\xi}{h_\eta} \nabla \xi \right) = -v_{ty} \frac{h_\xi}{h_\eta} \frac{\partial \xi}{\partial x} + v_{tx} \frac{h_\xi}{h_\eta} \frac{\partial \xi}{\partial y} = 0 \quad : \quad \Gamma_\eta \tag{23}$$

On the Γξ boundaries, the boundary condition of η is proposed to be

$$G_{n\eta} = \vec{u}_n \bullet \left(\frac{h_\eta}{h_\xi} \nabla \eta \right) = -u_{ty} \frac{h_\eta}{h_\xi} \frac{\partial \eta}{\partial x} + u_{tx} \frac{h_\eta}{h_\xi} \frac{\partial \eta}{\partial y} = 0 \quad : \quad \Gamma_\xi \tag{24}$$

The boundary conditions on Γξ given by (20) and (24) are

$$\begin{gathered} u_{tx} \frac{\partial \xi}{\partial x} + u_{ty} \frac{\partial \xi}{\partial y} = 0 \\ u_{tx} \frac{\partial \eta}{\partial y} - u_{ty} \frac{\partial \eta}{\partial x} = 0 \end{gathered} \tag{25}$$

Which requires

$$\frac{\partial \xi}{\partial x} \frac{\partial \eta}{\partial x} + \frac{\partial \eta}{\partial y} \frac{\partial \xi}{\partial y} = 0 \tag{26}$$

and is consistent with (17). The boundary conditions on Γη given by (21) and (23) are

$$\begin{gathered} v_{tx} \frac{\partial \eta}{\partial x} + v_{ty} \frac{\partial \eta}{\partial y} = 0 \\ v_{tx} \frac{\partial \xi}{\partial y} - v_{ty} \frac{\partial \xi}{\partial x} = 0 \end{gathered} \tag{27}$$

which requires.

$$\frac{\partial \xi}{\partial x} \frac{\partial \eta}{\partial x} + \frac{\partial \eta}{\partial y} \frac{\partial \xi}{\partial y} = 0 \tag{28}$$

and is also consitent with (17).

In summary, the physical plane grid generation problem requires the solution of

$$\frac{\partial}{\partial x}\left(\frac{1}{f}\frac{\partial \xi}{\partial x}\right)+\frac{\partial}{\partial y}\left(\frac{1}{f}\frac{\partial \xi}{\partial y}\right)=0$$

$$\xi=\xi_b \quad : \quad \Gamma_\xi \tag{29}$$

$$\vec{v}_n \bullet \left(\frac{1}{f}\nabla \xi\right)=0 \quad : \quad \Gamma_\eta$$

$$\frac{\partial}{\partial x}\left(f\frac{\partial \eta}{\partial x}\right)+\frac{\partial}{\partial y}\left(f\frac{\partial \eta}{\partial y}\right)=0$$

$$\eta=\eta_b \quad : \quad \Gamma_\eta \tag{30}$$

$$\vec{u}_n \bullet \left(f\nabla \eta\right)=0 \quad : \quad \Gamma_\xi$$

where

$$f=\frac{h_\eta}{h_\xi} \tag{31}$$

The grid generation equations (29) and (30) are nonlinearly coupled since the scale factors are a priori unknown and must be determined by

$$h_\xi=\sqrt{\left(\frac{\partial x}{\partial \xi}\right)^2+\left(\frac{\partial y}{\partial \xi}\right)^2}=\frac{\sqrt{\left(\frac{\partial \eta}{\partial x}\right)^2+\left(\frac{\partial \eta}{\partial y}\right)^2}}{\sqrt{\frac{\partial \xi}{\partial x}\frac{\partial \eta}{\partial y}-\frac{\partial \eta}{\partial x}\frac{\partial \xi}{\partial y}}} \tag{32}$$

$$h_\eta=\sqrt{\left(\frac{\partial x}{\partial \eta}\right)^2+\left(\frac{\partial y}{\partial \eta}\right)^2}=\frac{\sqrt{\left(\frac{\partial \xi}{\partial x}\right)^2+\left(\frac{\partial \xi}{\partial y}\right)^2}}{\sqrt{\frac{\partial \xi}{\partial x}\frac{\partial \eta}{\partial y}-\frac{\partial \eta}{\partial x}\frac{\partial \xi}{\partial y}}} \tag{33}$$

The nonlinear coupling requires an iterative solution scheme

$$\frac{\partial}{\partial x}\left(\frac{1}{f^{(n-1)}}\frac{\partial \xi^{(n)}}{\partial x}\right)+\frac{\partial}{\partial y}\left(\frac{1}{f^{(n-1)}}\frac{\partial \xi^{(n)}}{\partial y}\right)=0$$

$$\xi=\xi_b \quad : \quad \Gamma_\xi \tag{34}$$

$$\vec{v}_n \bullet \left(\frac{1}{f^{(n-1)}}\nabla \xi^{(n)}\right)=0 \quad : \quad \Gamma_\eta$$

$$\begin{gathered}\frac{\partial}{\partial x}\left(f^{(n-1)}\frac{\partial \eta^{(n)}}{\partial x}\right)+\frac{\partial}{\partial y}\left(f^{(n-1)}\frac{\partial \eta^{(n)}}{\partial y}\right)=0 \\ \eta=\eta_b \quad : \quad \Gamma_\eta \\ \vec{u}_n \bullet \left(f^{(n-1)}\nabla \eta^{(n)}\right)=0 \quad : \quad \Gamma_\xi \end{gathered} \tag{35}$$

Since the generation equations are to be solved in the physical domain, a spatial discretization capable of accurately implementing the mixed boundary conditions in (34) and (35) is required. At a given iteration level, (34) and (35) are analogous to heat conduction (or Darcian flow) in a heterogeneous media with mixed fixed temperature (pressure) and insulated (impermeable) boundary conditions. The finite element method is highly suited for this problem for a number of reasons. Since the zero normal gradient boundary conditions are natural boundary conditions for FEM solution of elliptic problems they are accurately implemented. Also, the complete grid generation problem requires location of (x,y) coordinates corresponding to specified (ξ,η) vertex values. The interpolation or basis functions of the finite element method provide a ready framework to accomplishing this.

Implementation of physical plane grid generation algorithm begins with idealizing the water body boundary, Figure 1, by a curvilinear block rectangular boundary, Figure 2, and a corresponding computational domain grid, Figure 3. Next, a quadralateral finite element grid is constructed in the physical domain. The grid in Figure 4, obtained by the solution of (11) and (12), readily defines the required finite element grid. The iterative generation equations (34) and (35), with f defined by (31), (32), and (33), are then solved on the finite element grid. Figures 7 and 8 show the specification of ξ boundary conditions for the solution of (34) and the η boundary conditions for the solution of (35), respectively. The resulting solution will have none integer values of (ξ,η) at the element nodes, the exception being the boundary nodes where integer values are assigned. The location of the (x,y) grid vertices corresponding to integer values of (ξ,η) spanning the range of the computational domain values is obtained by using bi-linear interpolation. Figure 7 shows the curvilinear grid generated by this approach. Note that the generation algorithm automatically distributes the computational vertices along the physical domain boundaries such that a high degree of numerical orthogonality is achieved. Figure 8 shows a different computational domain which also represents the physical domain in Figure 2. The grid corresponding to this computational domain is shown in Figure 9.

Appendix A References

Fletcher, C. A. J., 1991: Computational Techniques for Fluid Flow, Volume II. Springer-Verlag, Berlin.

Hamrick, J. M., 1996: Users manual for the environmental fluid dynamic computer code. The College of William and Mary, Virginia Institute of Marine Science, Special Report 328.

Knupp, P., and S. Steinberg, 1993: Fundamentals of Grid Generation. CRC Press, Boca Raton.

Mobley, C. D., and R. J. Stewart, 1980: On numerical generation of boundary fitted orthogonal curvilinear coordinate systems. . *Journal of Computational Physics*, **34**, 124-135.

Ryskin, G., and L. G. Leal, 1983: Orthogonal mapping. *Journal of Computational Physics*, **50**, 71-100.

Thompson, J. F., Z. U. A. Warsi, and C. W. Mastin, 1985: Numerical Grid Generation, Foundations and Applications. North-Holland, New York.

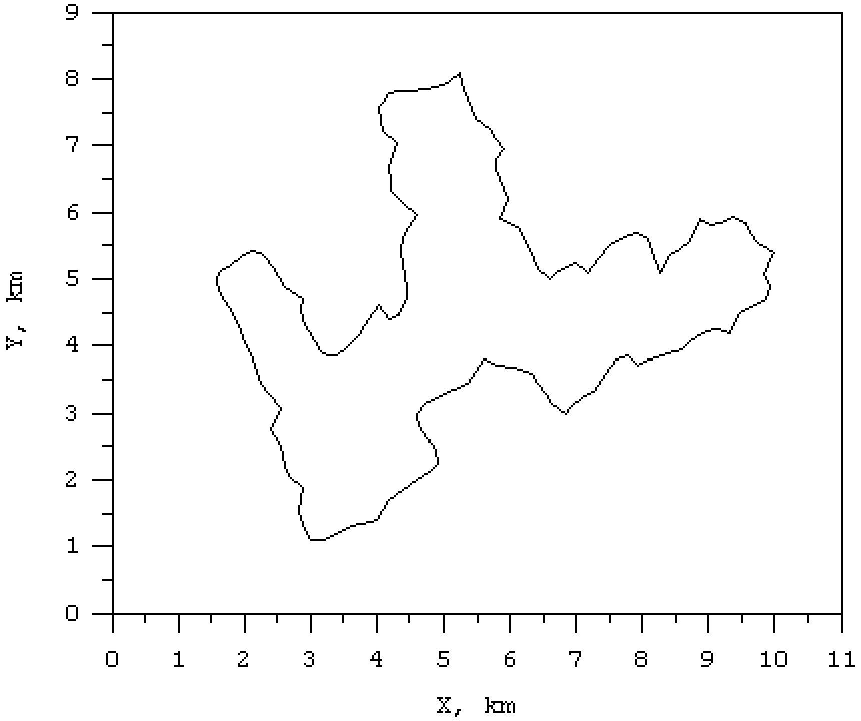


Figure A1. A water body shoreline in physical domain.

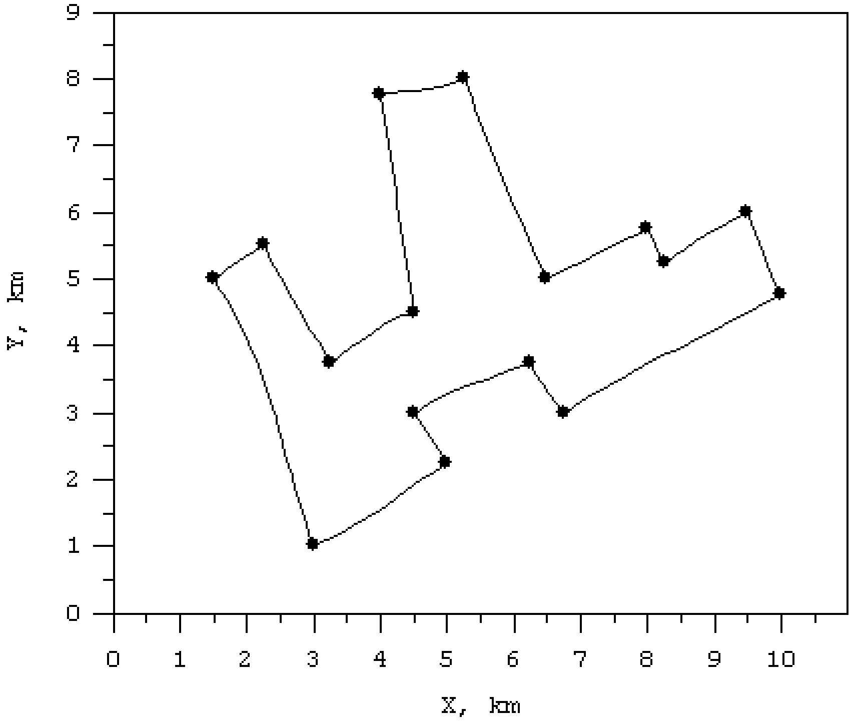


Figure A2. Approximation of water body shoreline in Figure 1 by a curvilinear block rectangular region in the physical domain.

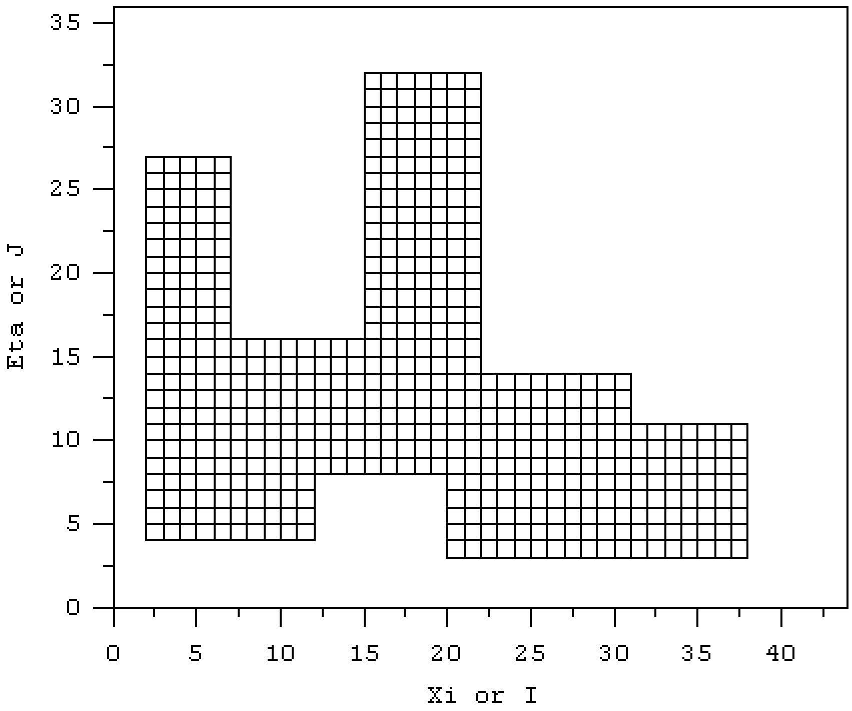


Figure A3. A computational domain corresponding to physical region in Figure 2.

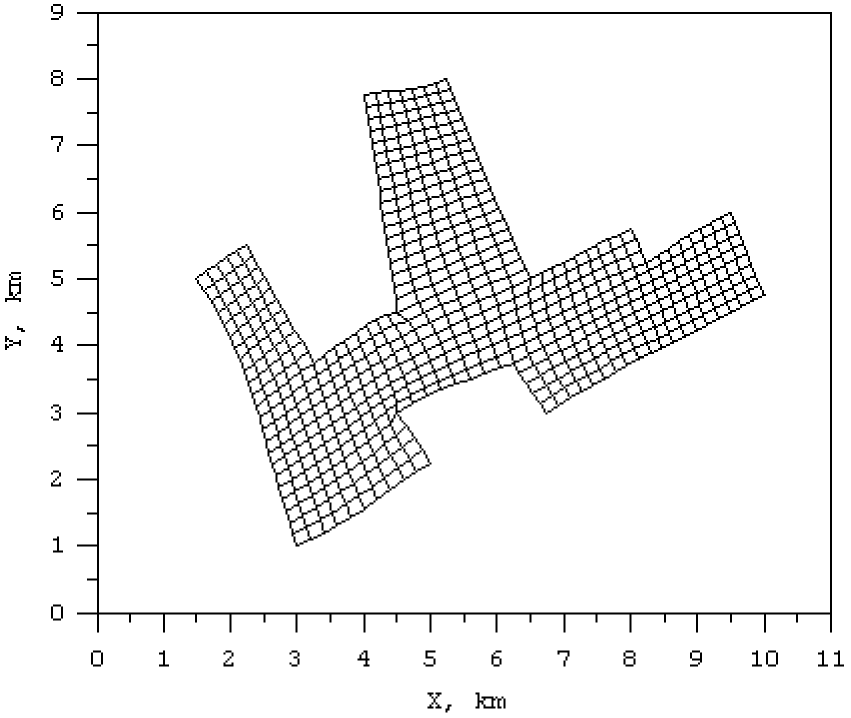


Figure A4. Initial grid corresponding to domains in Figures 2 and 3 determined by the solution of Equations (11) and (12).

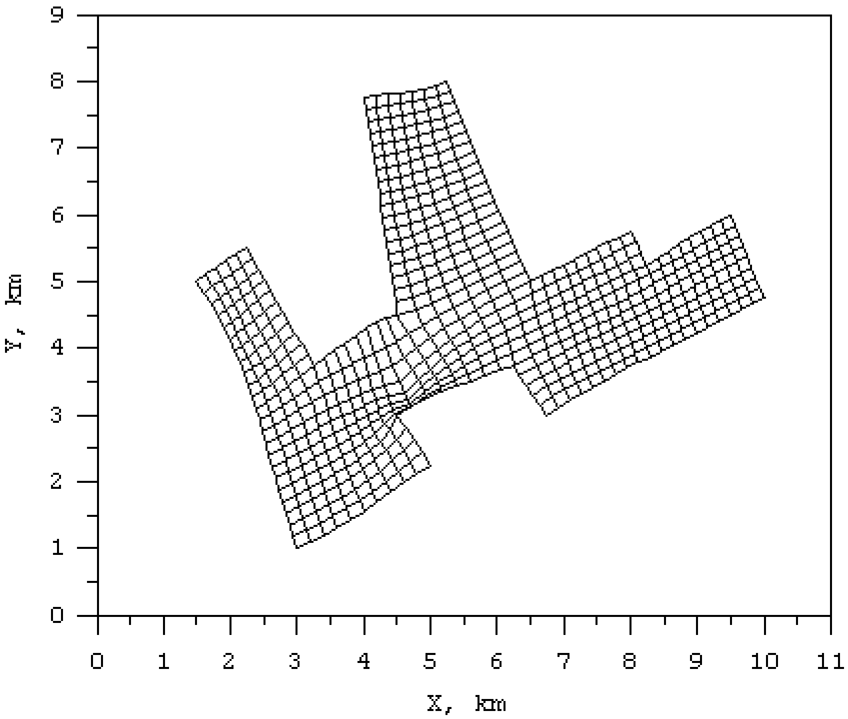


Figure A5. Grid determined by the solution of Equations (8) and (9) with defined approximately equal spacing of vertices along each boundary side.

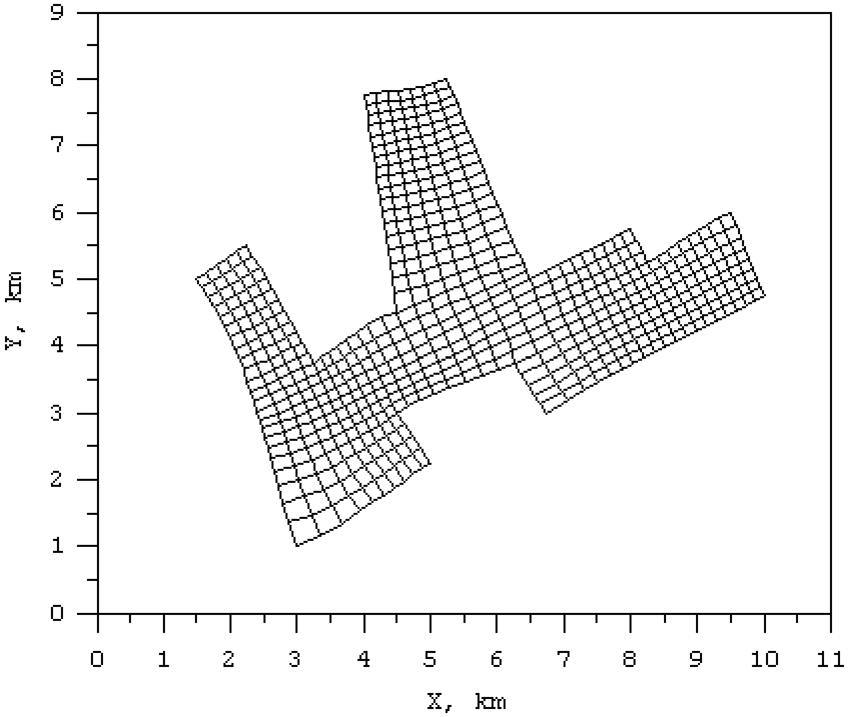


Figure A6. Grid determined by the solution of Equation (8) and (9) with defined variable spacing of vertices along each boundary side.

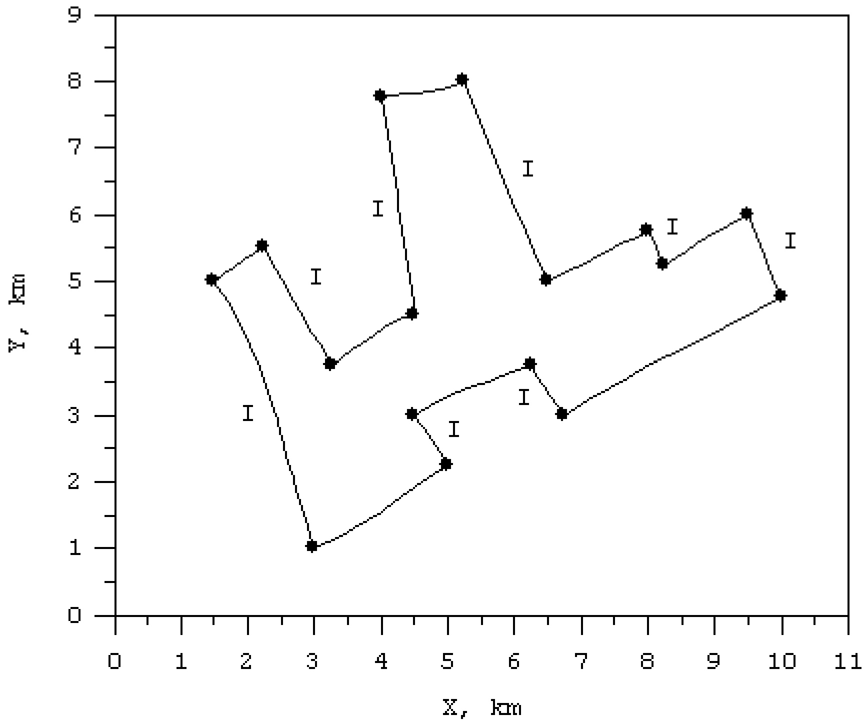


Figure A7. Boundary conditions for solution of equation (34) for ξ or I. Integer values of ξ specified on sides marked with I. Zero normal gradient specified on unmarked sides.

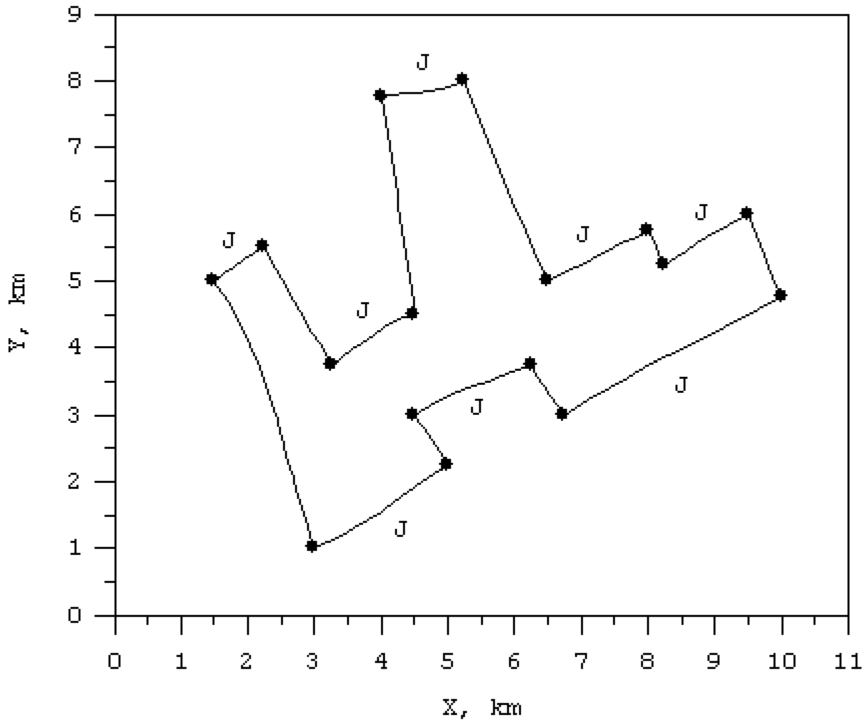


Figure A8. Boundary conditions for solution of equation (35) for η or J. Integer values of η specified on sides marked with J. Zero normal gradient specified on unmarked sides.

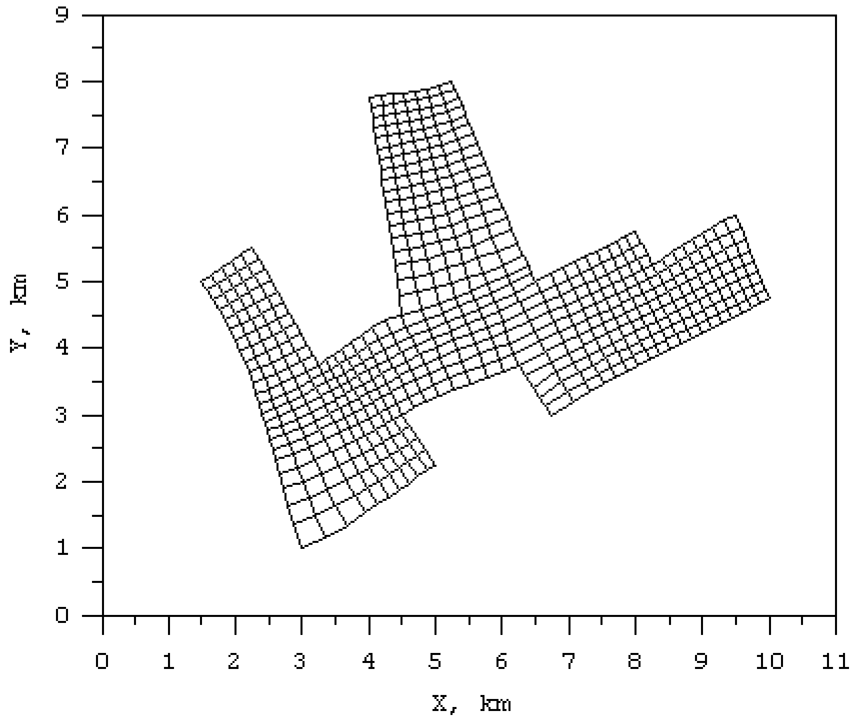


Figure A9. Grid determined by the solution of Equations (34) and (35), in the physical domain, with automatic spacing of vertices along each boundary side.

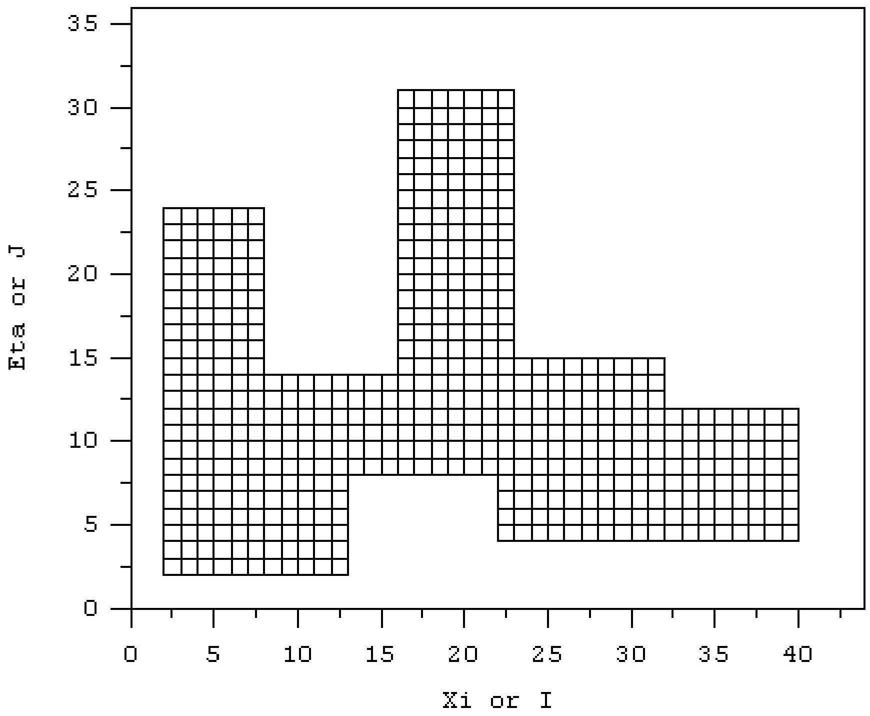


Figure A10. An alternate computational domain corresponding the physical domain in Figure A2.

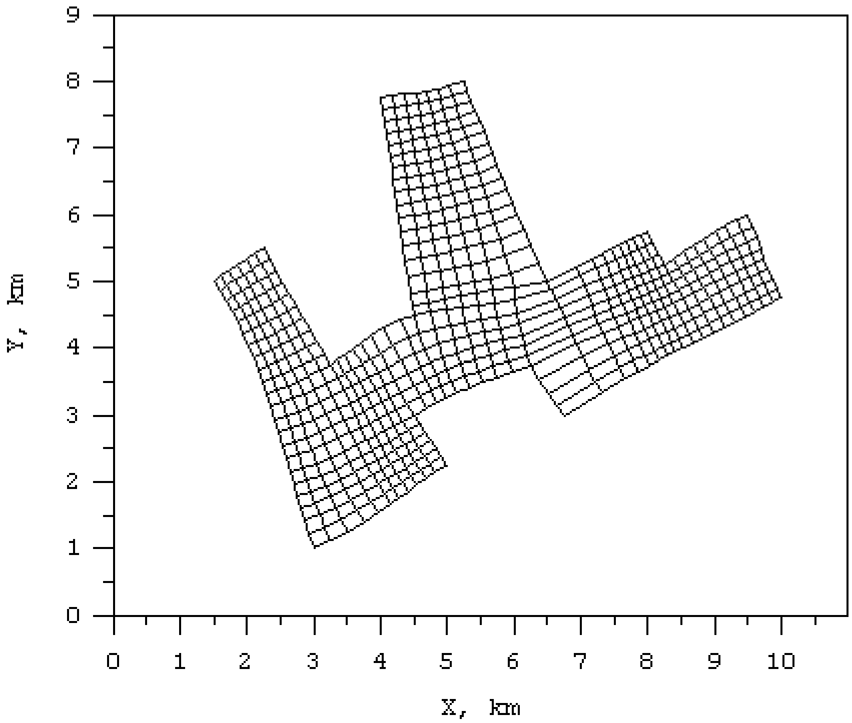


Figure A11. Grid determined by the solution of Equations (34) and (35) with automatic spacing of vertices along each boundary side corresponding the computational domain in Figure 10.

Appendix B: Shoreline and Bathymetry Conversion Programs

This appendix describes the use of the *fixshore.f* and *fixbath.f* FORTRAN programs for conversion of coastal shorelines and bathymetry into formats supported by VOGG and GRID. Coastal shoreline and bathymetry data are generally available in longitude-latitude format. In particular the NOAA National Geophysical Data Centers Coastal Relief Model CD's, Volumes 1 through 5 cover the US East and Gulf Coast.

The FIXSHORE program is designed to read extracted coastal shorelines that have been save in MAPGEN or ARC/INFO ungenerated format (.GEN). A typical saved shoreline data set containing mixed MAPGEN and GEN data is shown in Figure B1. The FIXSHORE program reads this data from the file having the generic name shore.raw. The FIXSHORE program also requires the control file *fixshore.inp* shown in Figure B2. Two output files are generated, shore.ll and shore.utm. Both files have three data columns, the first two being coordinate pairs (lon, lat) or (x,y) and the third being a pen lift control. The GRID interface requires shoreline data having geographic x and y coordinate in either meters or kilometers and can import the shore.utm file. The FIXSHORE program converts lon-lat coordinates in the input file to metric UTM coordinates for a user specified central meridian, rlonz. Local UTM coordinates relative to a lon-lat origin (rlono, rlato) can be selected and output in either meters or kilometers.

The FIXBATH program is designed to process coastal bathymetry data formatted in lon, lat, and depth. Depending upon the source of data, depth can be positive or negative. The NOAA Coastal Relief data model includes coastal land elevation data and denotes land by positive elevation and water by negative elevation, both relative to mean sea level. Figure B3 shores an example of the input file *bathdat.inp* containing the data. The VOGG program and GRID interface require depth data form the file depdat.inp which has data coordinates in the same coordinates as vertex coordinate and shoreline data. The FIXBATH program requires the input file *fixbath.inp*, which is identical to the *fixshore.inp* file with the exception that the number of data lines, ndat, is not required. The FIXBATH program outputs the *depdat.inp* required by VOGG for depth data interpolation.

```
#                          !start of mapgen format data
-82.498075     27.216467
-82.497855     27.216614
-82.498075     27.216614
-82.498075     27.216467
#
-82.498515     27.206786
-82.498515     27.206933
-82.498955     27.207520
-82.499029     27.208253
-82.499395     27.209060
-82.499762     27.208986
-82.499615     27.209280
-82.499982     27.210013
-82.500000     27.210013  !end of mapgen format data
1                         !start of gen format data
 -80.125114, 25.991419
 -80.125076, 25.991644
 -80.124832, 25.991642
 -80.124855, 25.991463
 -80.125114, 25.991419
END
2
 -80.126198, 25.989403
 -80.125076, 25.990091
 -80.125305, 25.990299
 -80.125458, 25.990671
 -80.125679, 25.990881
 -80.125465, 25.991602   !end of gen format data
```

Figure B1. Example of shoreline data read by FIXSHORE from file *shore.raw*.

```
c  cvtrlon    rlonz    rlono   rlato   scutm    ndat
   -1.0       81.      82.0    24.5    0.001    36


 cvtrlon = 1 if input longitude is positive and in west hemisphere
          -1 if input longitude is negative and in west hemisphere

 rlonz = positive west hemisphere central meridian for utm projection
         must be 3 + 6*n where n is a positive integer

 rlono = positive west hemisphere longitude for origin of local utm output
         coordinates

 rlato = positive north hemisphere latitude for origin of local utm output

 scutm = 1.0 for x, y output in meters
         0.001 for x, y output in kilometers

 ndat = number of lines in shore.raw input file (not read by fixbath)
```

Figure B2. Example of the fixhore.inp and fixbath.inp files.

```
-83.00000 27.00000 -33
-82.91667 27.00000 -29
-82.83333 27.00000 -28
-82.75000 27.00000 -21
-82.66667 27.00000 -15
-82.58333 27.00000 -12
-82.50000 27.00000 -9
-82.41667 27.00000 -6
-82.33333 27.00000 2
-82.25000 27.00000 -10
-82.16667 27.00000 4
-82.08333 27.00000 3
-82.00000 27.00000 8
-81.91667 27.00000 11
-81.83333 27.00000 18
-81.75000 27.00000 22
-81.66667 27.00000 21
-81.58333 27.00000 17
-81.50000 27.00000 16
-81.41667 27.00000 10
-81.33333 27.00000 8
-81.25000 27.00000 4
-81.16667 27.00000 -1
-81.08333 27.00000 -2
-81.00000 27.00000 -3
-80.91667 27.00000 –4
```

Figure B3. Example of the *bathdap.inp* file.