APPENDIX B

A Finite Element Solution Technique for a Diagnostic Circulation Model

by

G. Watabayashi and J. A. Galt

September 1978

TABLE OF CONTENTS

A FINITE ELEMENT SOLUTION TECHNIQUE FORA DIAGNOSTIC SHELF CIRCULATION MODEL

Glen Watabayashi and J. A. Galt

Pacific Marine Environmental Laboratory, ERL/NOAA
Seattle, Washington 98105

ABSTRACT. A linear diagnostic shelf circulation model developed by Galt (1975) is implemented using the Finite Element Method. The model solves a second-order non-homogeneous elliptic vorticity equation for the surface elevation within the region of interest. Solutions are obtained using finite element techniques, with elemental areas determined by available STD station spacing. After obtaining the surface elevations, velocities are calculated.

The model was initially tested on several simple contrived cases to help demonstrate the physics and the numerical techniques involved. Results from these tests indicate that, physically, the model generates a barotropic flow within the region of interest such that water and vorticity are conserved through the bottom Ekman layer. Numerically, the model approximates the analytical solution by piecewise linear functions. Therefore, if the anlaytical solution is not linear numerical errors occur which depend upon the mesh size.

The computer model has been written up in Standard Fortran and requires a set of STD station data and wind-stress data. The model is configured so that it can be economically run on intermediate size computers (100-150K core).

1. INTRODUCTION

The purpose of this study is to develop an economic and easily used flow model for continental shelf areas to help study the distribution of offshore pollutants. This report documents the program and demonstrates its use for simple test cases. A geostrophic model appropriately formulated for time scales of a few days is attempted. Most geostrophic flow models in the past have been developed for flow in deep water where a level of no motion is specified. At this reference level the net horizontal pressure gradient is assumed to be zero (Sverdrup et al., 1942; Formin, 1964). From this hypothesized level, the relative isopycnal slopes can be calculated from STD observations. Over shelf areas, however, a level of no motion is improbable, and a different kind of model is needed.

The model developed here for shelf areas is a linear steady-state model requiring a set of STD and wind-stress data. The model incorporates baroclinic contributions, a variable depth, a wind-driven surface Ekman layer, and a geostrophically driven bottom Ekman layer into a vertically integrated vorticity equation. Continuity is invoked; the coriolis parameter is taken to be a constant, and the final result is a nonhomogeneous elliptic equation for the surface elevation. (A similar formulation for the homogeneous case is presented by Welander, 1957.) This equation is solved using a finite element technique with a triangular mesh system which can be adjusted to a region of arbitrarily located stations. Once the surface elevations are obtained, estimates of surface and bottom velocities are computed.

The actual computer model calls a set of subroutines which can easily be bypassed, altered, or used elsewhere. For example, the model has a subroutine to convert geogrphic coordinates into a nondimensional x, y Cartesian grid, and another to normalize the raw station data in terms of arbitrary dimensions read in by the user.

This report will concentrate on the finite element technique used, the development of the mathematics and physics of the model has been published by Galt (1975). A companion report will discuss the details of the boundary conditions formulation and suggest strategies for model use.

2. THE FINITE ELEMENT TECHNIQUE

We now turn our attention to solving the elliptic model equation for the surface elevation. This equation is solved numerically using a finite element technique. The finite element approach copes with randomly spaced discrete data within a region, and the finite element grid fits odd-shaped regions well.

The finite element approach approximates the solution as a linear combination of "shape functions". These functions are inserted into the differential equation and the residual, or error, is minimized. For example, the equation to be solved is

$$N_2\nabla^2\xi - J(\xi, d) - N_1 J(\alpha, d) + N_1 N_2 \nabla^2\alpha - \nabla x \vec{\tau} = 0 \ . \tag{1}$$

This can be written in the following operator form

$$L(E) = F \ , \tag{2}$$

where

$$L(\xi) = N_2\nabla^2\xi - J(\xi, d) \ , \tag{3}$$

$$F = N_1 J(\alpha, d) - N_1 N_2 \nabla^2\alpha + \nabla x \vec{\tau} \ . \tag{4}$$

The solution, ξ, can be approximated in the following way:

$$\xi = \sum_{i=l}^{NVRTX} \psi_i C_i \ , \tag{5}$$

where

$NVRTX$ = number of points at which the equation will be solved (No. of stations);

$\psi_i = \psi_i(x, y)$, "shape function";

c_i = value of solution at specified locations,

The shape functions used here will be piecewise continuous and take on the value of one at node "i", and zero at neighboring nodes. The exact nature of the shape functions and the strategy behind them is explained in the following section. At this point, the ψ's are a linearly independent bases set of functions used to approximate the solution.

The next step is to substitute the approximate solution into equation (2) to give

$$L\left(\sum_{i=1}^{NVRTX} \psi_i c_i\right) - F = E , \qquad (6)$$

where

E = error introduced due to approximation.

The error is minimized by the Galerkin technique (Zienkiewicz, 1971). The method requires that the error be orthogonal to the space spanned by the bases set of functions. This is expressed by the following equation:

$$\iint_D E \cdot \psi_j dxdy = 0 \text{ for } j = 1, 2, \ldots . NVRTX, \qquad (7)$$

where

D = domain of interest.

Writing this as a system of NVRTX equations and substituting the expression for E from (6) into the above gives

$$\sum_{j=1}^{NVRTX} \iint_D \left(L\left(\sum_{i=1}^{NVRTX} \psi_i C_i \right) - F \right) \psi_j dxdy = 0 . \qquad (8)$$

Since the operator, L, is linear, and the C_i's are constants, this can be written as

$$\sum_{j=1}^{NVRTX} \sum_{i=1}^{NVRTX} C_i \iint_D L(\psi_i)\ \psi_j dxdy = \sum_{j=1}^{NVRTX} \iint_D F\psi_j dxdy \ . \qquad (9)$$

This can be written in matrix form as

$$AC = R , \qquad (10)$$

where

$$A_{ij} = \iint_D \psi_j L(\psi_i) dxdy , \qquad (11)$$

$$R_j = \iint_D F\psi_j dxdy . \qquad (12)$$

Substituting the operator from (3) into (11) gives

$$A_{ij} = \iint_D \left[N_2 \psi_j \left(\frac{\partial^2 \psi_i}{\partial x^2} + \frac{\partial^2 \psi_i}{\partial y^2} \right) - \psi_j \left(\frac{\partial \psi_i}{\partial y} \frac{\partial d}{\partial x} - \frac{\partial \psi_i}{\partial x} \frac{\partial d}{\partial y} \right) \right] dxdy . \qquad (13)$$

The shape functions, ψ, will be made up of linear functions of x and y, and are piecewise continuous across element boundaries. Thus the second derivative terms are not well-defined along the boundaries and the integration shown in equation (13) cannot be completed. To avoid this problem, the second derivative terms are integrated by parts (for details, see Appendix V), giving

$$A_{ij} = -N\iint_D \left(\frac{\partial\psi_i}{\partial x}\frac{\partial\psi_j}{\partial x} + \frac{\partial\psi_i}{\partial y}\frac{\partial\psi_j}{\partial y}\right) dxdy$$
$$-\iint_D \psi_j \left(\frac{\partial\psi_i}{\partial y}\frac{\partial d}{\partial x} - \frac{\partial\psi_i}{\partial x}\frac{\partial d}{\partial y}\right)dxdy + \oint_s \psi_j\left(\frac{\partial\psi_i}{\partial x}\ell y + \frac{\partial\psi_i}{\partial y}\ell x\right)ds \ , \tag{14}$$

where

s = boundary of the domain,

ℓ_x, ℓ_Y = directional cosines along the boundary.

Similarly, R_j may be obtained by substituting from (4) into (12):

$$R_j = \iint_D N_1\psi_j \frac{\partial\alpha}{\partial y}\frac{\alpha d}{\partial x} - \frac{\partial\alpha}{\partial x}\frac{\partial d}{\partial y} - N_1N_2\psi_j \frac{\partial^2\alpha}{\partial x^2} + \frac{\partial^2\alpha}{\partial y^2}$$
$$+ \psi_j\left(\frac{\partial\tau_y}{\partial x} - \frac{\partial\tau_x}{\partial y}\right)dxdy \ . \tag{15}$$

Once again integrating the second derivative term by parts gives

$$R_j = N\iint_D \psi_i\left(\frac{\partial\alpha}{\partial y}\frac{\partial d}{\partial x} - \frac{\partial\alpha}{\partial x}\frac{\partial d}{\partial y}\right)dxdy + N_1N_2\iint\left(\frac{\partial\psi_j}{\partial x}\frac{\partial\alpha}{\partial x} + \frac{\partial\psi_j}{\partial y}\frac{\partial\alpha}{\partial y}\right)dxdy$$
$$+ \iint_D \psi_j\left(\frac{\partial\tau_y}{\partial x} - \frac{\partial\tau_x}{\partial y}\right)dxdy - N_1N_2\oint\psi_j\left(\frac{\partial\alpha}{\partial x}\ell_y + \frac{\partial\alpha}{\partial y}\ell_x\right)ds \ . \tag{16}$$

Equations (14) and (16) now define the matrix equations which must be solved.

Now consider the geometrical problem of calculating A_{ij} and R_j. First the domain is divided into triangular vertices. The five-station case is an example in figure 1. At each station, the position, depth, a, A, and wind stress components are given as

s(N), Y(N), depth (N), alpha (N), delta (N), taux (N), tauy (N),

where N refers to the global label of the station,

Within each triangle, the bases functions and independent variables are all assumed to be linear functions of x and y. This means that within each triangle the independent variables are represented as

$$depth_{TN} = D_x \cdot x + D_y \cdot y + D_o ,$$
$$alpha_{TN} = \alpha_x \cdot x + \alpha_y + y + 0 ,$$
$$taux_{TN} = \tau_x \cdot x + \tau_x \cdot y + \tau_{x\,o} ,$$
$$tauy_{TN} = \tau_y \cdot x + \tau_y \cdot y + \tau_{y\,o} .$$

The coefficients are determined by matching values at the vertices. For example, to solve for D_x, D_y, and Do in triangle T1, we solve the following set of equations:

$$depth\ (1) = D_x \cdot x(I) + D_y \cdot y(I) + Do ,$$
$$depth\ (III) = D_x \cdot x(111) + D_y \cdot y(III) + D_o ,$$
$$depth\ (V) = Ox \cdot x(V) + D_y \cdot y(V) + D_o .$$

Each triangle contributes to the value of the shape function at each of its vertices. For example, triangle T1 contributes to the value of the shape function of points I, 111, and V. The contributing elements to the shape functions are defined as follows:

$$
\begin{aligned}
\psi_I^{TN} &= \psi_x(1) \cdot x + \psi_y(1) \cdot y + \psi_0(1) , \\
\psi_J^{TN} &= \psi_x(2) \cdot x + \psi_y(2) \cdot y + \psi_0(2) , \\
\psi_K^{TN} &= \psi_x(3) \cdot x + \psi_y(3) \cdot Y + \psi_0(3) .
\end{aligned}
\qquad (17)
$$

where

TN = triangle number,

I, J, K = vertex number.

The coefficients, ψ_x, ψ_y, and ψ_0 are determined in such a way that ψ_I^{TN} · 1 at I and zero at vertices J and K. As an example, in triangle Tl, to obtain $\psi_x(1)$, $\psi_y(1)$, and $\psi_0(1)$, the following set of equations is solved:

$$
\begin{aligned}
1 &= \psi_x(1) \cdot x(1) + \psi_y(1) \cdot Y(1) + \psi_0(1) , \\
0 &= \psi_x(1) \cdot x(2) + \psi_y(1) \cdot Y(2) + \psi_0(1) , \\
0 &= \psi_x(1) \cdot x(3) + \psi_y(1) \cdot Y(3) + \psi_0(1) .
\end{aligned}
\qquad (18)
$$

Within each triangle the shape functions and independent variables are planar segments. Over the entire region, the shape functions and independent variables are piecewise continuous.

The next step is to assemble the matrix and right-hand side of equation (10) one triangle at a time. Since the independent variables and shape function are linear, all the first derivatives are constants. Therefore, A_{ij} can be rewritten as

$$A_{ij} = - N_2 \iint_{DTN} (\psi_x(K_i) \cdot \psi_x(K_j) + \psi_y(K_i) \cdot \psi_y(K_j) dxdy \qquad (19)$$

$$- \iint_{DTN} \psi_i (\psi_y(K_i) \cdot D_x - \psi_x(K_i) \cdot D_y) + \oint \psi_j (\frac{\partial \xi_i}{\partial x} \ell_y + \frac{\partial \xi_i}{\partial y} \ell_x) ds ,$$

where

DTN = Domain of triangle TN,

K_i, K_j = refers to the global coefficient associated with points i and j.

R_j becomes

$$N_1 \iint_j (\alpha_y D_x - \alpha_x \cdot D_y) dxdy + N_1 N_2 \iint_{DTN} ((\psi_x)_j \alpha_x + (\psi_y)_j \alpha_y) dxdy$$

$$+ \iint_{DTN} (\psi_j (\tau_y)_x - (\tau_x)_y) dxdy - N_1 N_2 \int_S \psi_j (\alpha_x \cdot \ell_y + \alpha_y \ell_x) ds. \qquad (20)$$

Notice that the line integrals contribute only to the points which lie on the boundary of the entire domain. In the five-point example, the line integrals are zero unless both i and j do not equal V.

To evaluate A_{ij} and R_j, three types of integrals have to be evaluated for each triangle:

(a) $\iint_{DN} dxdy$,

(b) $\iint_{DTN} \psi dxdy$,

(c) $\oint_{S} \Psi ds$.

The first is simply the area of the triangle. The second is one-third the area of the triangle (see Appendix VI), and the third equals the length of the triangle sides adjacent to the boundary point divided by two. A_{ij} and R_j can now be rewritten as:

$$A_{ij} = - N_2(\psi_x(K_i) \cdot \psi_x(K_j) + \psi_y(K_i) \cdot \psi_y(K_j)) \cdot \text{Area} \tag{21}$$

$$- (\psi_y(K_i) \cdot D_x - \psi_x(K_i) \cdot D_y) \cdot 1/3 \text{ Area}$$

$$+ (\psi_x(K_i) \cdot \ell_y + \psi_y(K_i) \cdot \ell_x) \cdot \oint \psi_j ds ,$$

$$R_j = N_1(\alpha_y \cdot D_x - \alpha_x \cdot D_y) \cdot 1/3 \text{ Area} + N_1 N_2(\psi_x(K_j) \cdot ax \tag{22}$$

$$+ N_1 N_2(a_x \cdot \ell_y + \alpha_y \cdot \ell_x) \oint \psi_j ds .$$

The matrix and right-hand side are now ready to be assembled by adding the contributions from each triangle. In our test case, we begin with triangle T1. All the gradients are calculated along with the area of the triangle. Then the contributions to R_j and A_{ij} are calculated and placed in their appropriate locations. For Tl, i = 1, III, V, and j = I, III, V; the contributions to A_{ij} would be at A_{11}, A_{13}, A_{15}, '31' '33' '35' '51' '53', and A_{55}, while the contributions to R_j would be to R_1, R_3, and R_5. The line integral terms contribute only when i and j are boundary points. After T1 is completed, the system is repeated for T2. For the second triangle A_{15} and A_{51} already have values from the previous triangle, so we add on to the existing values.

After all the triangles are covered, the boundary conditions are considered. The solution vector components c_i are the surface elevations at the triangle vertices which are known along the boundary. To incorporate the boundary conditions where the elevation is given, the rows associated with the boundary points are set to zero except for the diagonal element which is set to 1. Then the element of the right-hand side associated with this row is set to the boundary value. Along island or coastline boundaries a no net transport condition is added on to the assembled matrix. Along these boundaries we require:

$$-d\frac{\partial \xi}{\partial s} + N_2\left(\frac{\partial \xi}{\partial n} - \frac{\partial \xi}{\partial s}\right) = - N_1\left(\frac{\partial \Delta}{\partial s} + \alpha\frac{\partial d}{\partial s}\right) + \tau_s - N_1N_2\left(\frac{\partial \alpha}{\partial n} - \frac{\partial \alpha}{\partial s}\right) \qquad (23)$$

Where $\bar{n}$ is a unit vector normal to the coast pointing offshore and $\bar{s}$ is a unit vector given by $\bar{k} \times \bar{n} = \bar{s}$, where $\bar{k}$ is positive up. To see how this is incorporated into the assembled matrix consider the following triangle with a coastal boundary (figure 3)

We see:

$$\bar{n} = n_x \bar{i} + n_y \bar{j} \tag{24}$$

$$= \left[\frac{(y_m - y_\ell)}{\left[(x_m - x_\ell)^2 + (y_m - y_\ell)^2 \right]^{\frac{1}{2}}} \right] \bar{i} + \left[\frac{-(x_m - x_\ell)}{\left[(x_m - x_\ell)^2 + (y_m - y_\ell)^2 \right]^{\frac{1}{2}}} \right] \bar{j}$$

$$\bar{s} = s_x \bar{i} + s_y \bar{j} \tag{25}$$

$$= \left[\frac{(x_m - x_\ell)}{\left[(x_m - x_\ell)^2 + (y_m - y_\ell)^2 \right]^{\frac{1}{2}}} \right] \bar{i} + \left[\frac{(y_m - y_\ell)}{\left[(x_m - x_\ell)^2 + (y_m - y_\ell)^2 \right]^{\frac{1}{2}}} \right] \bar{j}$$

Within this triangle all variables are expressed in terms of the three shape functions, i.e.,

$$\xi = C_\ell \psi_\ell + C_m \psi_m + C_n \psi_n$$

$$d = d_\ell \psi_\ell + d_m \psi_m + d_n \psi_n$$

$$a = \alpha_\ell \psi_\ell + \alpha_m \psi_m + \alpha_n \psi_n$$

$$A = \Delta_\ell \psi_\ell + \Delta_m \psi_m + \Delta_n \psi_n$$

And the shape functions are defined by

$$\psi_\ell = (\psi_\ell)_x \, x + (\psi_\ell)_y \, y + (\psi_\ell)_o$$

$$\psi_m = (\psi_m)_x \, x + (\psi_m)_y \, y + (\psi_m)_o$$

$$\psi_n = (\psi_n)_x \, x + (\psi_n)_y \, y + (\psi_n)_o$$

With these normal and tangential derivatives can be defined by

$$\frac{\partial \xi}{\partial n} = (C_\ell (\psi_\ell)_x + C_m (\psi_m)_x + C_n (\psi_n)_x)\ n_x + (C_\ell (\psi_\ell)_y + C_m (\psi_m)_y + C_n (\psi_n)_y)\ n_y \tag{26}$$

Using these forms we can substitute into equation (23) to get a relationship between known triangle parameters and the nodal values of the dependent variable. The error in this equation is then required to be orthogonal to the bases set of functions integrated along the coastal or island boundary. These constraints are added on to the matrix which has already been assembled using the differential equation.

Details are shown in the program listing given in Appendix II.

3. THE PROGRAM

The FORTRAN program making up the model satisfies several specifications. First and foremost, the program can be easily utilized by anyone who has a set of standard STD station data and an available computer. Second, the program has several options as to what is read, computed, and printed. Third, parts of the program are easily changeable, bypassed, or omitted without affecting other parts of the program. The program is basically written as a collection of overlays and subroutines. In the main program, the user specifies what type of data is to be read, what is to be computed, and what is to be printed. The main program subsequently activates the appropriate set of overlays and subroutines.

Before dealing with the program in detail, it would be helpful to briefly summarize the program. It begins by reading in several control parameters which dictate what is to be computed, listed, punched, and plotted. The program has the option to list whatever is read and computed, and to punch whatever is computed. This allows data to be easily echo-checked, and computed values need not be recomputed for future runs using the same source deck. The first set of control parameters deals with normalizing the station data. If raw station data is read in with corresponding geographic coordinates, the program will transform the positions onto a scaled x-y Mercator grid and normalize the station data according to scale parameters which are also read in. The normalized data can be punched onto cards for later runs. The next set of options concerns the triangular mesh used for interpolation and as the finite element mesh. The user has the option of either reading in the triangular mesh or using a set of subroutines in the program to create the triangles. Then the boundary values are read in. If the triangles are internally generated, the triangles external to the region are eliminated. The program proceeds to generate and solve the finite element matrix and right-hand side vector subject to the boundary conditions. The solution yields the surface elevation at each station, and with this information, the transport, mean velocity, surface slope velocity, wind-driven surface velocity, and the geostrophic velocity at the bottom for each triangle are calculated. Finally, there is a set of plotting option which will draw and contour the results.

Appendix I is a flow chart of the main program and overlay structure with explanations of the key routines. A listing of the complete program is given in Appendix II.

3. Documentation

3.1 Section I

The program begins by reading the control parameters. These parameters determine what the program will do and how it will function. There are three types of control parameters. The first allows the user to bypass an option. For example, if NOGRID is set to O, the program will not generate a Mercator grid; instead, it will read the grid. The second type of control parameter allows one to list whatever is read or calculated. These parameters all begin with the letter L. For example, if LTRI is set to 1, the program will list the triangle vertices. The third type of control parameter begins with the letters 1P and determines if the program will punch the results on cards. For example, if IPNORM is set to 1, the program will punch the normalized station data. A detailed explanation of each option is given in the program itself (see Appendix II). If any control parameter is set to 1, the option will be executed and, if it is O, the option will be bypassed. In addition to the list, punch, and bypass options, there are parameters which allow the user to alter the boundary conditions during the given run, store the decomposed matrix on a file for later use, and smooth the alpha and delta field to a least squares fit over the data.

3.2 Section II

This section deals with the input of station data. There are several options available. The first is to set NORMAL and NOGRID to 1 and read the raw station data. The program will then generate a Mercator grid and normalize the station data. Another choice is to read in normalized station data with geographic coordinates, or raw station data with Mercator coordinates. In the last two cases, either NOGRID or NORMAL is set to O. The last option is to read normalized station data with Cartesian coordinates for the station locations. In this case, both NOGRID and NORMAL are set to O. If more than one run is made on the same set of data, the last option should be exercised after generating a data deck of the normalized data and Mercator grid from the initial run.

Raw station data is read using format 10. An example is given in Appendix 121.

3,4 Section 111

The scale parameters which control the scaling and nondimensional izing are read in here, they are as follows:

USCALE: Velocity scale in meters per second;

DSCALE : Depth scale in meters;

ALSCALE: Horizontal length scale in meters;

G : Gravity in meters per second squared;

E : Perturbation density in grams per centimeter cubed;

Q : Constant density in grams per centimeter cubed;

GAMMA : Bottom friction coefficient in grams per centimeter squared.

3.5 Section IV

In this section, geographic coordinates are transformed into Mercator x-y grid. Notice that the routine works only for the northern hemisphere and west longitude. The subroutine finds the maximum and minimum values of latitude and longitude. The minimum latitude becomes the y = 0 axis. The y coordinate value of each station is obtained by calculating its distance from the y = 0 axis and scaling the distance by the horizontal length scale. For example, if a station is 100 km north of the y = 0 line and the length scale is 100 km, the y coordinate value of the station is one unit. The x coordinate is computed in a slightly different manner because the distance between a station and the x = 0 line is a function of its longitude and latitude. The program finds the mean latitude and calculates the distance in the x direction that the station is from the x = 0 longitude and the longitude of the station at the mean latitude.

The Mercator grid can be scaled so the output overlays standard hydrographic charts. The transformations are as follows:

$$X(I) = -(R * \lambda - \lambda_{min})$$

$$Y(I) = R * ALOG(TAN(\emptyset/2. + \pi/4) - TAN(\frac{\emptyset_{min}}{2} + \pi/4))$$

where:

R = Radius of Earth + (ALSCALE *COS (Ø average)) this makes each nondimensional unit in the horizontal direction one scale (ALSCALE) length at the mean latitude.

λ = Longitude

λ_{min} = Western most longitude to become x = o line

$\emptyset$ = Latitude

$\emptyset_{min}$ = minimum latitude to become y = o line

The radius, R, is listed by the routine and if a plot is to be made to fit another Mercator projection, the x and y axis can be scaled appropriately. For example on a typical Mercator chart there will be a statement that the scale of the projection is 1:N on reference latitude θ ref. Then if R is entered as:

$$R = \frac{\text{Radius of the earth (meters)}}{} * \cos(\emptyset_{REF})$$

The plots will be correctly scaled to overlay the chart.

The data for this routine is read with the raw station data. The degrees of latitude and longitude are read into two integer arrays and the minutes of latitude and longitude into two decimal places are read into two real arrays. The x and y coordinates returned from this subroutine are stored into the real arrays, and the original latitude and longitude of the stations are lost.

3.6 Section V

This routine calculates the mean coriolis value, FO, by averaging the maximum and minimum coriolis values of the region.

3.7 Section VI

The raw station data is normalized according to the scale parameters read in by Section III. In the following, the primed values are the normalized, nondimensional values:

```
ALPHA' = (ALPHA - Q* DEPTH)/(E * DSCALE);
DELTA' = (DELTA-Q* DEPTH *DEPTH/2 )/( E* DSCALE *DSCALE);
DEPTH ' = DEPTH/DSCALE
TAUX ' = (1/(FO * USCALE * Q * DSCALE)) * TAUX;
TAUY ' = (1/(FO * USCALE * Q * DSCALE)) * TAUY;
CURL ' = (1/(FO * USCALE * Q * DSCALE * ALSCALE)) * CURL,
```

The normalized values are stored in the arrays where the raw data was stored.

3.8 Section VII

In this section, a least squares fit is made over the alpha and delta fields. A third-order polynomial in the z direction is fit to both the alpha and delta data.

```
ALPHA = A. + A1*Z + A2*Z*Z + A3*Z*Z*Z
DELTA = Do + D1*Z + D2*Z*Z + D3*Z*Z*Z
```

The coefficients are returned by the subroutine to be used later to calculate the alpha and delta gradients. The subroutine is a general least squares fit program, and the basis set of functions used for the least squares interpolation can be changed. The user also has the option of smoothing the alpha and delta fields. For example, if the parameter “SMOOTH” is set to 1.5, the program will smooth data over 1.5 standard deviations away from the least squares fit back to 1.5 standard deviations.

3.9 Section VIII

In this section, the nondimensionalized run parameters are either read in or calculated:

CONST1 = (G * E * H)/(Q * FO * USCALE * ALSCALE);

CONST2 = GAMMA/(Q * DSCALE).

3.10 Section IX

Here the program either reads in the triangles or generates them. The triangle information is stored in an array, IP(I, N), where N is the number of the triangle and I is the local vertex number (1, 2, or 3). The value of 1P is the global number of the particular point. See figure 4 for an example.

If the triangles are to be read in, integer format 5 is used (see Appendix III for an example). A brief explanation of how the triangles are generated is given in Appendix IV.

3.11 Section X

The boundary values are read in this section. If the triangles are generated internally, the external boundary values must be read in counter-clockwise order. This will allow the program to determine what triangles are inside or outside of the domain. The boundary values needed are the surface elevations of the boundary stations in centimeters, Coastal and island boundary points must be identified. A discussion of the strategy used to obtain boundary values is given in the comparison report in this series. See Appendix 111 for an example of how the boundary values should be read.

3.12 Section XI

This section is used only if the triangles are generated internally. The subroutine FINDBP stores the number of boundary points that each triangle has, rearranges the local vertex numbers of the boundary triangles so the boundary points are the lead vertices in counterclockwise order, and finally checks to see that the boundary points are ordered consistently. This section sets up the boundary triangles for the next routine which eliminates the triangles external to the region (see figure 5).

3.13 Section XII

Again, this option is needed only if the triangles are generated internally. Subroutine ELIM is used to eliminate the triangles external to the region. This will be the case for concave domains (see figure 6) and islands. Each triangle with three boundary points is tested to see if it is external or internal to the region (see figure 6).

The final mesh of the triangles to be used for the finite element technique and the number of triangles, NTRI, are products of this section.

3.14 Section XIII

Subroutine SETMAT zeros the global matrix and right-hand side. The subroutine also has the option of eliminating the trianlges with three boundary points from the finite element mesh. If the second option is activated, the last parameter in the call, IELI, is set to 1.

3.15 Section XIV

The assembly of the global matrix starts here. K is the number of the triangle being operated on.

3.16 Section XV

The triangle vertices are identified in terms of their global labels. The first vertex of triangle Khas global label J, the second, L, and the third, M.

3.17 Section XVI

The three-by-three location matrix is set and used to calculate the area of the triangle and all the gradients within the triangle. For example, the first row of the matrix contains the x and y coordinates of the first local vertex of triangle K.

3.18 Section XVII

Now the area of the triangle is calculated. Subroutine TRIAREA calculates the determinant of the location matrix A, and multiplies it by a half. The absolute value of this quantity becomes the area of the triangle.

3.19 Section XVIII

The gradients needed for the triangle are calculated with the exception of the alpha gradients. The alpha gradients are calculated in the next section. The other forcing function gradients are obtained here using the position matrix as a coefficient matrix and setting the right-hand side vector, B(l), B(2), B(3), equal to the particular values of the forcing function at vertices 1, 2, and 3. For example, for the depth we have:

$$\begin{bmatrix} \\ A \\ \\ \end{bmatrix} \begin{bmatrix} \frac{d}{x} \\ \frac{d}{y} \\ d_o \end{bmatrix} = \begin{bmatrix} B(1) \\ B(2) \\ B(3) \end{bmatrix} \begin{matrix} = \\ = \\ = \end{matrix} \begin{bmatrix} \text{Depth at vertex 1} \\ \text{Depth at vertex 2} \\ \text{Depth at vertex 3} \end{bmatrix}$$

The three-by-three system is solved by Kramer's Rule in subroutine SOLVE which calls TRIAREA to compute the determinants.

The shape function gradients are also calculated in the same manner by solving the following set of equations:

$$\begin{bmatrix} \\ A \\ \\ \end{bmatrix} \begin{bmatrix} \frac{d\ shape\ (I)}{dx} \\ \frac{d\ shape\ (I)}{dy} \\ c\ shape\ (I) \end{bmatrix} = \begin{bmatrix} B(1) \\ B(2) \\ B(3) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}_{I=1} \text{ or } \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}_{I=2} \text{ or } \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}_{I=3}$$

The subscript I tells you which vertex the particular gradient is associated with.

3.20 Section XIX

The alpha gradients are calculated differently than the other gradients. A simple linearization of the alpha field introduces errors which are unacceptably large, so a more detailed description of the density field is needed than the bottom alpha values at the triangle vertices. Therefore, a third-order least squares fit to the alpha field is generated (see Section VII) and used to obtain alpha values at the triangle vertices for the centroid depth.

$$\text{Alpha (at centroid depth)} = \text{alpha (at bottom)} + \frac{\partial(\text{alpha})}{\partial z} * \delta z + \frac{\partial^2(\text{alpha})}{\partial z^2} * \frac{\delta Z^2}{2} + \frac{\partial^2(\text{alpha})}{\partial^3} * \frac{\partial Z^3}{6}$$

In the above calculations, the alpha gradients are obtained by differentiating the least squares function of alpha. Once the alpha values at the centroid depth are obtained over each vertex, subroutine GRAD is called to calculate the horizontal alpha gradients. The delta gradients needed for the transports are calculated in the same manner,

3.21 Section XX

The triangle's contribution to the global matrix and right-hand side is added in here. Each triangle contributes to particular rows and columns of the global matrix determined by the global label of the triangle vertices (figure 7).

$$\begin{array}{cc} & \begin{matrix} 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} \end{array} \begin{bmatrix} \\ \epsilon \\ \\ \end{bmatrix} = \begin{bmatrix} x \\ x \\ x \end{bmatrix} \begin{matrix} 3 \\ 4 \\ 5 \end{matrix}$$

Then the second triangle would contribute to

$$
\begin{array}{c}
 \\ 3 \\ 5 \\ 6
\end{array}
\quad
\begin{array}{c}
\begin{array}{ccc} 3 & 5 & 6 \end{array} \\
\begin{bmatrix} + & + & + \\ + & + & + \\ + & + & + \end{bmatrix}
\end{array}
\begin{bmatrix} \\ C \\ \\ \end{bmatrix}
=
\begin{bmatrix} + \\ + \\ + \end{bmatrix}
\quad
\begin{array}{c}
3 \\ 5 \\ 6
\end{array}
$$

Since we want the final global matrix to represent equation 36 integrated over the entire domain, we add all the contributions from each triangle. When contributions from triangles I and II in the example above are added, the resultant matrix looks like:

$$
\begin{array}{c}
 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6
\end{array}
\quad
\begin{array}{c}
\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & * & x & * & + \\
0 & 0 & x & x & x & 0 \\
0 & 0 & * & x & * & + \\
0 & 0 & + & 0 & + & +
\end{bmatrix}
\end{array}
\begin{bmatrix} \\ \\ \\ C \\ \\ \\ \end{bmatrix}
=
\begin{bmatrix} 0 \\ 0 \\ * \\ x \\ * \\ + \end{bmatrix}
\quad
\begin{array}{c}
1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6
\end{array}
$$

As the integration is done triangle by triangle, contributions to the matrix are accumulated in the appropriate locations of the global matrix and right-hand side.

3.22 Section XXI

The boundary conditions are imposed upon the solution in this section. Each row in the global matrix that is associated with a boundary point is zeroed. Then the diagonal element of that row is set to 1, and the element in the right-hand side associated with that row is set equal to the boundary value. For example, suppose a four-by-four system of equations was assembled as shown:

$$\begin{vmatrix} GM(1,1) & GM(1,2) & GM(1,3) & GM(1,4) \\ GM(2,1) & GM(2,2) & GM(2,3) & GM(2,4) \\ GM(3,1) & GM(3,2) & GM(3,3) & GM(3,4) \\ GM(4,1) & GM(4,2) & GM(4,3) & GM(4,4) \end{vmatrix} \begin{bmatrix} C(1) \\ c(2) \\ c(3) \\ c(4) \end{bmatrix} = \begin{bmatrix} RHS(1) \\ RHS(2) \\ RHS(3) \\ RHS(4) \end{bmatrix}$$

Then suppose that C(1) and C(2) are known boundary values. Subroutine BC alters the system into the following:

$$\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ GM(3,1) & GM(3,2) & GM(3,3) & GM(3,4) \\ GM(4,1) & GM(4,2) & GM(4,3) & GM(4,4) \end{vmatrix} \begin{bmatrix} C(1) \\ c(2) \\ c(3) \\ c(4) \end{bmatrix} \quad \begin{bmatrix} BV(1) \\ BV(2) \\ RHS(3) \\ RHS(4) \end{bmatrix}$$

Coastal and island boundary vertices have a no net flux boundary constraint added to the assembled matrix.

3.23 Section XXII

The system of equations is now solved by subroutine SOLN. This matrix solving routine programmed by Steve Smyth from Knuth (1968, 1973) takes advantage of the sparseness of the matrix by not storing zero values, and a second to tell us where in the matrix the nonzero values occur. The program is set to solve a 200 by 200 system with each row containing no more than 20 nonzero elements and another array, INTP (4900) keeps track of where the nonzero elements belong in the matrix. If a larger matrix is to be solved, the two arrays can be increased in a manner described within the program itself, The routine begins by reducing the global matrix into the product of an upper triangular and lower triangular system. Partial pivoting on the columns is used and the triangular matrices are stored into the original global matrix. The lower triangular system is solved first, then the upper triangular system is solved for vector C.

$$\begin{bmatrix} GM \end{bmatrix} \quad \text{becomes} \quad \begin{bmatrix} & U \\ L & \end{bmatrix}$$

The system now can be written as

$$\begin{bmatrix} & 0 \\ L & \end{bmatrix} \begin{bmatrix} & U \\ 0 & \end{bmatrix} \underset{\sim}{C} = R\underset{\sim}{H}S \ .$$

Let

$$\begin{bmatrix} & U \\ 0 & \end{bmatrix} \underset{\sim}{C} = \underset{\sim}{y} \ .$$

now solve the following system:

$$\begin{bmatrix} 1 \end{bmatrix} Y = R\underset{\sim}{H}S \ .$$

Once y is obtained, the following system is solved for C:

$$\begin{bmatrix} & 0 \\ U & \end{bmatrix} \underset{\sim}{C} = \underset{\sim}{y} \ .$$

The decomposed global matrix and right-hand side are saved. If different sets of boundary conditions are to be tested, the same decomposed global matrix is used and the right-hand side is adjusted accordingly.

3.24 Section XXIII

Here surface velocities and bottom geostrophic velocities are calculated for the centroid of the triangles.

3.25 Section XXIV

This is an option to calculate the terms of the vorticity equation. The values are calculated at the triangle centroids and are as follows:

Barotropic torque = $J(\xi, d)$,

Baroclinc torque = $N_1 J(\alpha, d)$,

Wind stress = $\nabla \times \vec{\tau}^{W}$,

Bottom friction = -(barotropic torque + barocl ini c torque + wind stress).

3.26 Section XXV

The plotting is executed in this section. The plotting is basically handled by several subroutines which draw and label the triangles, label the vertices, and contour any parameter defined at the vertices. A separate program is used to take the punched velocity data from the model and plot velocity arrows at the centroids of the triangle.

3.27 Section XXVI

This is the option to alter the right-hand side of the global system of equations to take into account new boundary conditions. The user determines the input for this subroutine for each set of boundary conditions. Basically, the routine changes the specific boundary values in the right-hand side vector. Once this is done, the program returns to SOLN and resolves the system of equations using the decomposed global matrix, subject to the new boundary conditions. With this system, numerous sets of boundary conditions can be tested at minimal cost.

4. RESULTS

Once the theory and software for the model was developed, operational testing was carried out. The model was run on simple contrived test cases. These runs were made to develop a better understanding of the physics involved to test the finite element method which is regularly used in engineering studies but is relatively new to oceanography. A summary of the results along with a discussion of the problems encountered will be presented here.

4.1 Test Cases

The test cases were designed to give a clear idea of how the model reacts to different physical conditions. Four simple cases were analyzed. In the first two, the finite element technique yielded exact linear solutions. In the last two cases, the analytical solutions could not be exactly represented by the first-order bases set and the accuracy of the numerical solution depended upon the resolution yielded by the mesh system.

The first test case was run on a regular six by six grid with a mesh of 50 triangles. The boundary elevations increased uniformly to the north from zero to 5 cm and the wind stress was zero. The depth and density were constant, and the nondimensional parameters were: CONST1 = 1.00, and CONST2 = .025. The vorticity equation reduces to Laplace's equation subject to the linear boundary conditions. Physically, the geostrophic flow is forced onlyby the surface slope and is unidirectional and nondivergent. The geostrophically driven bottom Ekman layer is also nondivergent and is transporting water from north to south. The analytical solution to the vorticity equation is $\xi = ky$ where k is a constant determined by the boundary elevation slope. The numerical solution for this case is exact since the finite element solution approximate the solution with piecewise linear functions and is accurate to the first order (fig. 8).

The second test case is nearly identical to the first. The boundary conditions were the same and once again, there was no stratification or wind forcing. For this case, however, the depth was decreased uniformly toward the north from 1200 m to 200 m.

In this case the vorticity equation becomes

$$N^2 \nabla^2 \xi + \frac{\partial d}{\partial y} \frac{\partial \xi}{\partial x} = 0 ,$$

and the resulting geostrophic flow is unchanged from the previous case. The surface slope once again drives a unidirectional, nondivergent current which has no shear except in the bottom Ekman layer. Mass and vorticity are conserved within the region by having the geostrophic flow follow isobaths. The solution once again is $\xi = ky$, and the numerical solution is exact (fig. 9).

In the third case, baroclinicity was introduced into the model by taking the density as a linear function ofy. The bottom depth, wind stress and boundary conditions were identical to those of case 2. Alpha, the integrated density, became a second-order function of y. The vorticity equation for this case reduces to:

$$N_2\nabla^2\xi + \frac{\partial d}{\partial y}\frac{\partial \xi}{\partial x} = N_1N_2\nabla^2\alpha = \text{const.}$$

Now, the linear basis set of functions used to approximate the solution cannot fit the exact solution and numerical errors are expected. Physically, the density field, depth, and boundary conditions are only functions of y and the resulting barotropic and baroclinic flows are in the x direction and nondivergent. The baroclinic mode increases with depth and flows counter to the barotropic mode. This results in a level of no motion at the mean depth of 700 m. Above the level of no motion, the boundary forced barotropic mode dominates, and the gesotrophic flow is to the west. This in turn drives a bottom Ekman layer to the south. Below the level of no motion, the baroclinic mode dominates? and the geostrophic flow is to the east. This forces a bottom Ekman layer to the north (Fig. 10). Therefore, the bottom Ekman layer forced by the boundary conditions and baroclinic field is convergent. However, the total flow must be nondivergent, and the interior barotropic mode (specified by the dependent variable, surface height) must adjust over the prescribed bathymetry to compensate for the bottom Ekman convergence. In seeking the analytic solution, we first note the similarity between the reduced voriticity equation for this case and Stommel 's model equation (1965).

Stommel 's Equation

$$\nabla^2\psi + \frac{D}{R}B\frac{\partial \psi}{\partial x} = \sin\frac{\tau y}{b}$$

$\psi = 0$ on Boundary

Diagnostic Model, Case 3 Equation

$$\nabla^2\xi + \frac{\partial d}{\partial y}\frac{\partial \xi}{\partial x} = N_1\nabla^2\alpha$$

$\xi = K_y$ on Boundary

This is a consequence of the integrated friction term being set proportional to the velocity in both models and the bathymetric stretching term for this case being of the same form as Stommel's beta term. If the boundary conditions for our vorticity equation were homogeneous, we would then expect the solution to be of the same form as Stommel's solution. The total solution in this case is just a linear combination of the solutions for the homogeneous equation solved for the nonhomogeneous boundary conditions (case 2; see equation A below) and the nonhomogeneous equation solved for the homogeneous boundary conditions (Stommel) type solution; (see equation B below).

Homogeneous Equation

$$N_2\,\nabla^2\xi + \frac{\partial d}{\partial y}\,\frac{\partial}{\partial x} = 0$$

$\xi = k_y$ on Boundary

Nonhomogeneous Equation

$$N_2\nabla^2\xi + \frac{\partial d}{\partial y}\frac{\partial \xi}{\partial x} = N_1 N_2\ \nabla^2 a$$

B

$\xi = 0$ on Boundary

Note the effect of the baroclinicity is to add a secondary flow onto the results obtained from the constant density case. Physically we can expect the solution to show southward flow into deeper water to compensate for the converging bottom Ekman layer and then flow moving back to the north along the western boundary to satisfy the boundary conditions. The results of the total solution show wave-like oscillations which are unrealistic and not what was expected from the analytical solution. Haney (1975) describes similar oscillatory solutions when a numerical mesh cannot resolve a boundary layer. 10 see if this is the case, the secondary flow is examined by subtracting the results of the homogeneous equation subject to the nonhomogeneous boundary conditions, (i. e., case 2) from the total solution. Figure 11 indicates that the mesh system may have problems resolving the secondary flow. A western boundary current structure is evident but not clearly resolved. To show that this is the problem, the boundary layer size was increased by setting the nondimensional friction parameter, CONST2, equal to 1.25 from its original .025.

In case 3A the secondary flow in figures 12 and 13 is now well resolved. As expected, the secondary flow resembles Stommel's solution with a western boundary barotropic mode forcing water to the south to compensate for the convergent bottom Ekman layer and then moving back to the north along the Western boundary to satisfy the imposed boundary conditions. This confirms that the problem with the original solution for the third case was related to resolving the boundary layer.

The next step was to see if the triangles along the western boundary could be cut in half to increase the resolution of the secondary flow. The previous case (case 3) was run again on the same mesh with CONST2 equal to .25. From the results in Figures 14 and 15, it can be seen that the numerical grid does not clearly resolve the boundary layer.

For case 3B, the mesh system was then altered such that the triangles along the western boundary were halved. The results are shown in Figure 16.

The solution is improved for Case 3C. It is more symmetrical and the surface elevation gradients are not as large. In general, a decrease in mesh size leads to more resolution, but the improvement is difficult to quantify because the finite element solution depends upon the triangle shapes as was the mesh size. For example, as the triangles become less equilateral, the global matrix becomes less conditioned (Strang and Fix, 1973).

For the fourth case, the density is once again set to a constant. The wind stress is still zero and the boundary condition is once again linear in the y direction. These are the same conditions as in the first case, but now the depth is made to be a linear function of both x and y (depth = Ax + By+ C). See Figure 17.

The vorticity equation becomes:

$$N_2 \nabla^2 \xi + \frac{\partial d}{\partial y} \frac{\partial \xi}{\partial x} - \frac{\partial d}{\partial x} \frac{\partial \xi}{\partial y} = 0$$

The depth gradients are constants. Once again the linear basis set of functions used to approximate the solution cannot fit the exact solution and numerical errors are expected. In the interior, the flow attempts to follow isobaths but most deviate from the isobaths near the boundaries due to the boundary conditions. Along the north-south boundaries, the barotropic mode is not allowed to force water into or out of the region, while the shallower water on the eastern boundary allows the barotropic mode to force less water into the region than is leaving on the western boundary. To compensate for this, a secondary flow with a boundary layer on the western side is set up. The secondary flow forces a convergent bottom Ekman layer to conserve water within the region.

In Figures 18 and 19 of the solution for case 4, clearly, the secondary flow and its boundary layer are not well-resolved. As in the previous case, the boundary layer thickness is increased by increasing the friction coefficient, CONST2, by an order of magnitude to .25.

In Figures 20 and 21, the solution for case 4A showing surface elevation and secondary flow, the large oscillations are gone and it is clear that the flow attempts to follow an isobath until it reaches the western boundary. The counterclockwise secondary flow and its western boundary layer is now well-defined. It forces a convergent bottom Ekman layer which compensates for the excess water the barotropic mode forces out of the region through the western boundary.

The next step is to increase the resolution along the western boundary by once again halving the triangle size along the western boundary.

In Figures 22 and 23 for case 4B, the difference in the solution yielded by the two different meshes is almost negligible. The current along the western boundary is better resolved, but the finer mesh results in only a slightly more symmetrical solution. The probable reason for this is that water is converging along the northern boundary and diverging along the southern boundary, and to improve the solution, more resolution along these boundaries is needed.

The results can be summarized by saying that the model physically compensates for continuity mismatches between the surface Ekman layer, boundary forced barotropic flow, and within the region which has either a convergent or divergent bottom Ekman layer. This secondary flow is similar to Stommel 's model (1965) with a western boundary current and is a consequence of setting the vertically integrated friction terms proportional to the velocity. The model's inability to resolve the secondary flow due to too coarse a mesh was a problem in the test cases. Halving the mesh size along the western boundary improved the solution, but triangle shape as well as size affected the numerical solution.

The last test case indicates that care should be taken in setting the boundary conditions. If unrealistic boundary conditions are imposed, the model will compensate by forming a boundary layer which may degrade the results. The boundary layer thickness depends on the potential vorticity gradient (i.e., bottom slope) so the problem will be different for different geophysical settings.

5. CONCLUSION

A diagnostic shelf circulation model developed by Galt (1975) is implemented using the finite element method. The model is quasi-geostrophic and incorporates variable depth, baroclinicity, a surface Ekman layer, and a bottom Ekman layer. Physically, the model assumes a steady state, a small Rossby number flow. The depth scale is taken to be much less than the horizontal length scale, and the bottom Ekman layer is assumed to be driven by a geostrophic flow. The coriolis parameter is set to a constant and vorticity balance is required between the barotropic and baroclinic stretching terms and the bottom and surface Ekman layers. The test cases indicate that the model accommodates the boundary conditions and forcing functions by creating a secondary barotropic flow within the region to conserved mass and vorticity through the bottom Ekman layer.

The model solves an integrated vorticity equation which is a second-order, nonhomogeneous, elliptic equation and is tested subject to Dirichlet boundary conditions. The dependent variable is the surface elevation solved for by the Finite Element Method. The program is written in Standard Fortran and is a collection of subroutines and overlays which can be easily altered, bypassed or used elsewhere. The input data requires standard STD station data, wind stress information, and the boundary surface elevations.

The major problems the model encounters are numerical. The spatial resolution of the model is limited and the exact position of current features cannot be predicted to any greater accuracy than the available input data. This means that although the model clearly recognizes the local dynamics, its resolution with respect to position, is no better than the station spacing, and this should be taken into consideration prior to taking stations. The stations must be spaced to create a mesh which can resolve both the secondary flow and forcing functions, particularly the density field, and depth.

ACKNOWLEDGEMENTS

The work described in this report has been sponsored in part by the Outer Continental Shelf Environmental Assessment Program under RU#140.

REFERENCES

Formin, L. M., (1964): The dynamic method in oceanography. Elsevier, New York, 211 pp.

Galt, J. A., (1975): Development of a simplified diagnostic model for the interpretation of oceanographic data, NOAA Technical Report ERL 339-PMEL 25, National Oceanic and Atmospheric Administration, U.S. Department of Commerce, Washington, D.C., 36 pp.

Haney, R.L., and Wright, Jr. J.M., (1975): The relationship between the grid size and the coefficient of nonlinear lateral eddy viscosity in numerical ocean circulation models. J. Comp. Phys., 19, 257-266.

Knuth, Donald E. (1968): The Art of Computer Programming, Vol. 1, Fundamental Algorithms. Addison-Wesley Publishing Company, Menlo Park, 634 pp.

Knuth, Donald E., (1973): The Art of Computer Programming, Vol. 3, Sorting and Searching, Addison-Wesley Publishing Company, Menlo Park, 723 pp.

Stommel, H., (1965): The Gulf Stream. University of California Press, Berkeley, 248 pp.

Strang, G., and Fix, G., (1973): An analysis of the finite element method. Prentice Hall, Englewood Cliffs, N.J., 306 pp.

Sverdrup, H.U., Johnson, M.W., and Fleming, R.H., (1942): The oceans: Their physics, chemistry and general biology. Prentice Hall, Englewood Cliffs, N.J., 1087 pp.

Welander, P., (1957): Wind action on shallow sea: Some generalization of Ekman's Theory. Tellus, IX, 45-52.

Zienkiewicz, O.C., (1971): The Finite Element Method in Engineering Science, McGraw-Hill, London, 521 pp.

FIGURE CAPTIONS

Figure

#1 Five Station case where Roman numerals are global labels and triangles are labeled T1 to T4.

#2 Illustration of a piecewise continuous hat function associated with node V.

#3 Boundary triangle showing normal and alongshore directions.

#4 Sample triangle with labeling.

#5 Example of boundary triangles with labeling. Number of boundary points for each triangle:

IBTRI (1) = 2

IBTRI (2) = 1

IBTRI (3) = 2

Relabeled local vertices so that boundary points lead in a counter-clockwise order:

IP (1,1) = 40;	IP (1,2) = 41;	IP(1,3) = 41
IP (2,1) = 41;	IP (2,2) = 43;	IP(2,3) = 42
IP (3,1) = 44;	IP (3,2) = 44;	IP(3,3) = 43

#6 Example of a triangle external to the region of interest. If the outward normal to the boundary is in the triangle, then the triangle is external to the region. In this example the outward normal between vertices 1 and 2 fall inside triangle I so triangle I is outside of the domain and" eliminated.

#7 Two triangle examples.

#8 Homogeneous water, flat bottom case, where the geostrophic flow is

#8 (cont) nondivergent and toward the west and the bottom Ekman flow is nondivergent and toward the east.

#9 Homogeneous water, sloping bottom case, where the geostrophic flow is nondivergent and toward the west and the bottom Ekman" flow is nondivergent and toward the south.

#10 Baroclinic case with density and depth uniformly increasing toward the south and a level of no motion at the mean depth above the level of no motion.

#11 Secondary flow for case 3 where the density is a linear function of y. The boundary layer is not resolved, resulting in numerical oscillation. Elevations are in centimeters.

#12 Surface elevation contours for case 3 with boundary layer thickness increased (CONST2 = 1.25). Contours are in centimeters.

#13 Secondary flow for case 3 with boundary layer thickness increased. Western boundary layer is now well resolved. Elevations are in centimeters. (CONST2 = 1.25).

#14 Surface elevations for case 3 with CONST2 = .25. The boundary layer is not clearly resolved. Contours are .1 centimeter.

#15 Secondary flow for case 3 where CONST2 = .25. The boundary layer is not clearly resolved. Contours are .1 centimeter.

#16 Secondary flow for case 3 where the triangles along the western boundary are halved from previous case. The North-South symmetry of boundary layer is not yet fully resolved. Elevations are in centimeters. (CONST2 = .25)

#17 Non-dimensional isobaths for case 4 where each unit is equivalent to 200 m.. Depth = Ax + By + C.

#18 Surface elevations for case 4 where contours are in centimeters. The oscillations indicate that the boundary layer is not resolved.

#19 Secondary flow for case 4 where contours are in centimeters and the boundary layer is not well resolved.

#20 Surface elevations for case 4 where contours are in centimeters. CONST2 = .25,

#21 Secondary flow for case 4 with CONST2 = .25. Elevations are in centimeters.

#22 Surface elevations for case 4 with CONST2 = .25 and the triangles along the western boundary are halved.

#23 Secondary flow for case 4with CONST2 = .25 and the triangles a'long the western boundary have been halved from the previous case (figure 14). Elevations are in centimeters.

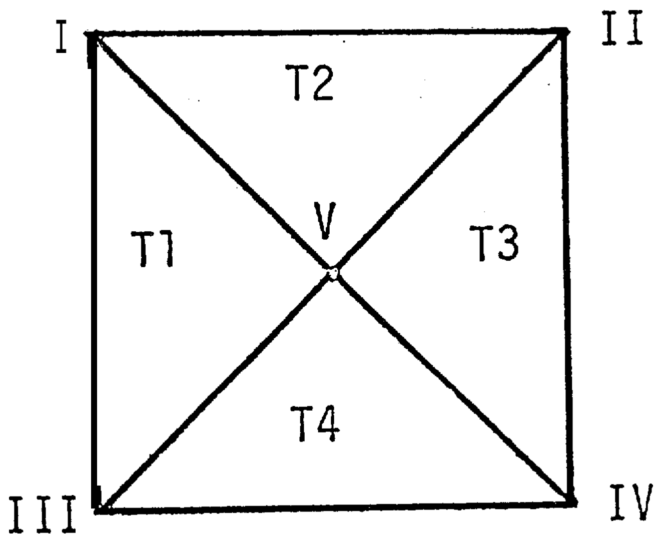


Figure 1.

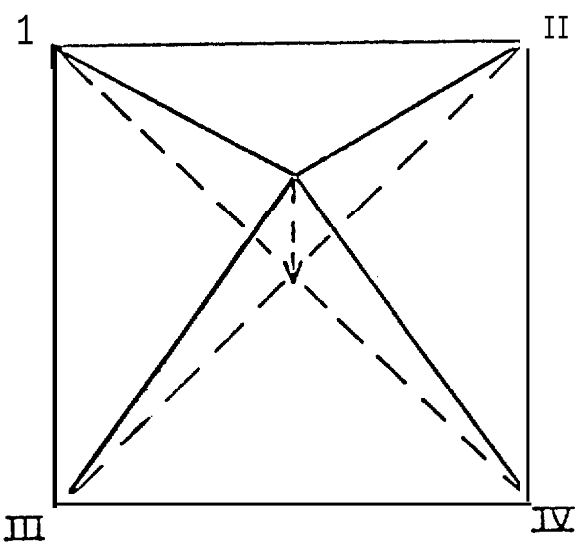


Figure 2.

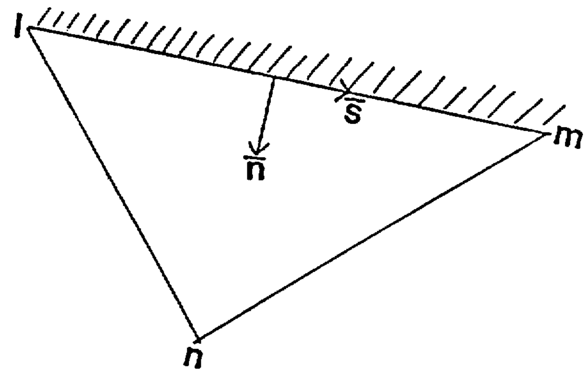


Figure 3.

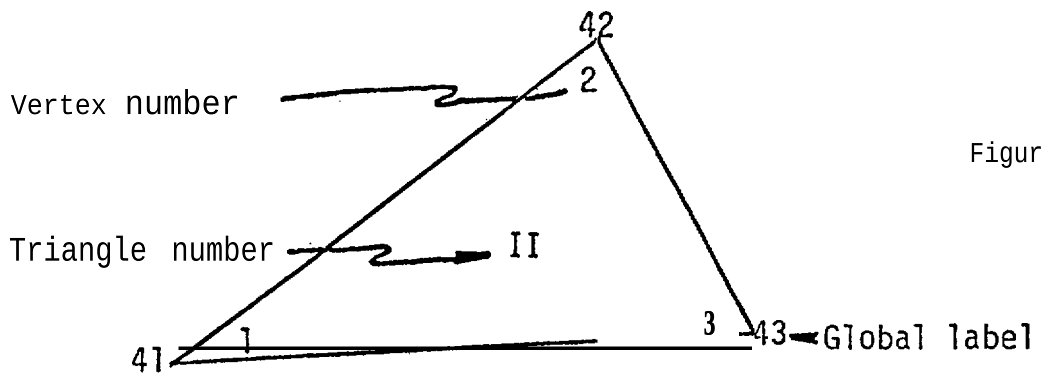


Figure 4.

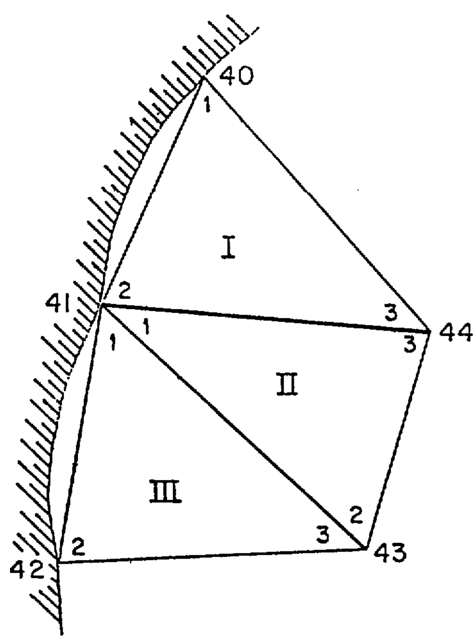


Figure 5.

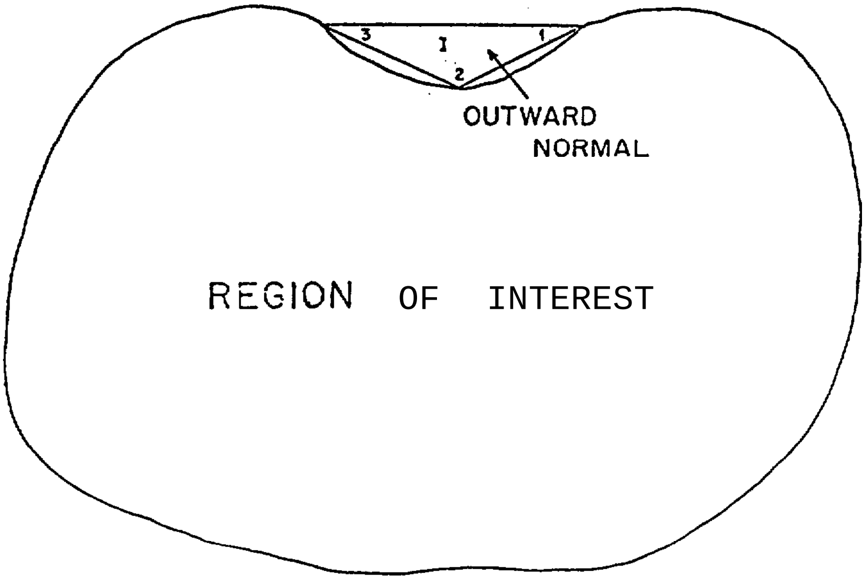


Figure 6.

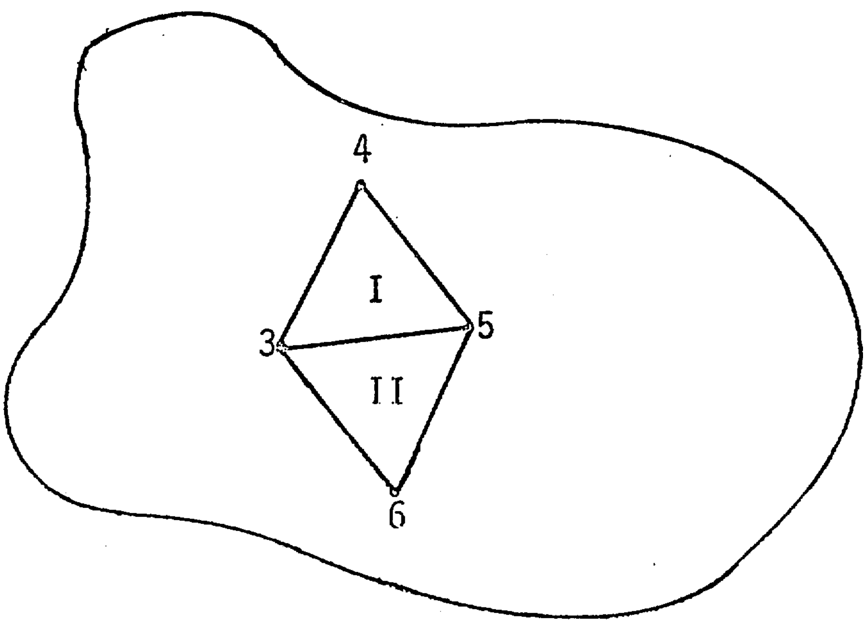


Figure 7.

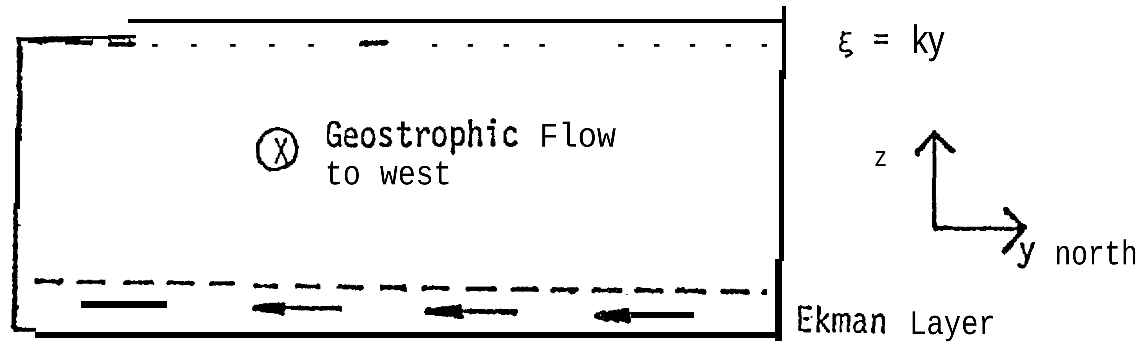
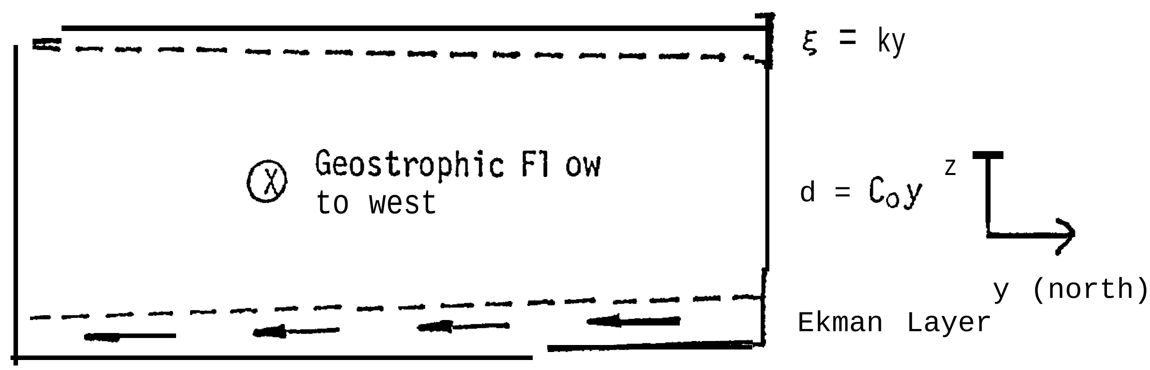


Figure 8.



F gure 9.

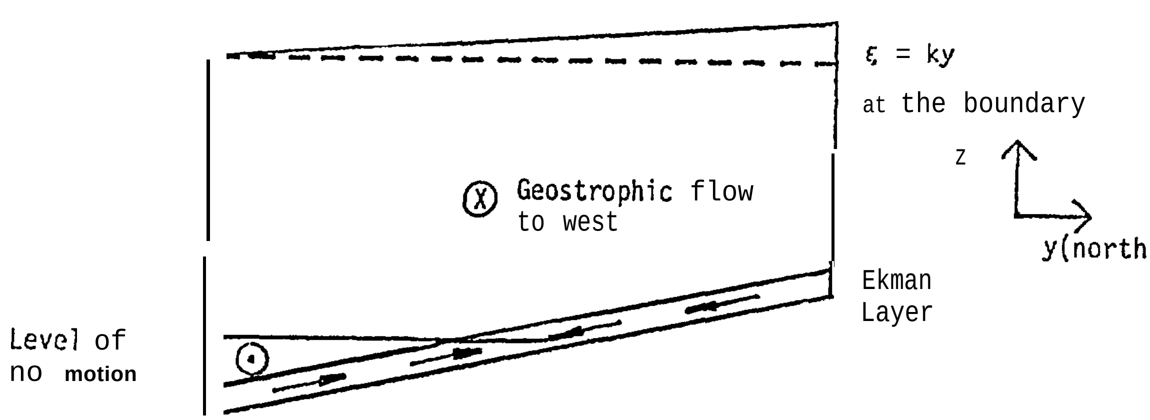


Figure 10.

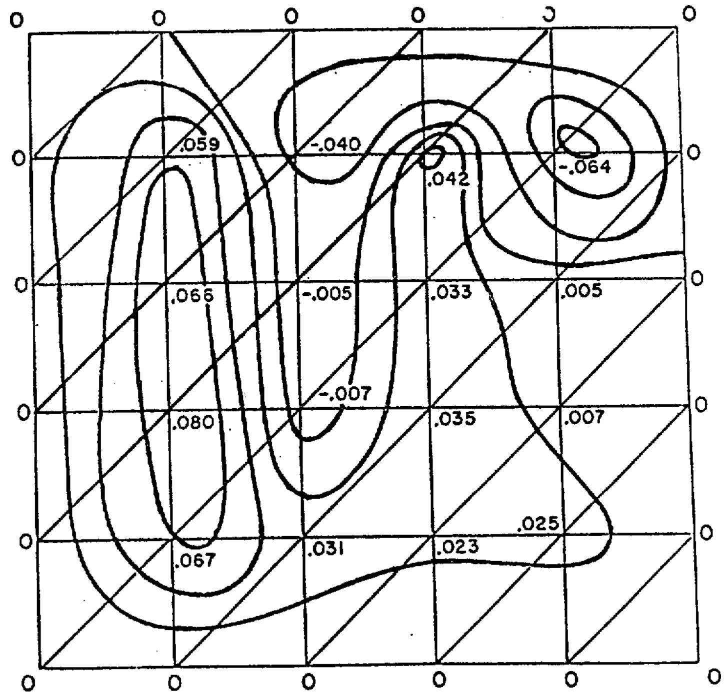


Figure 11.

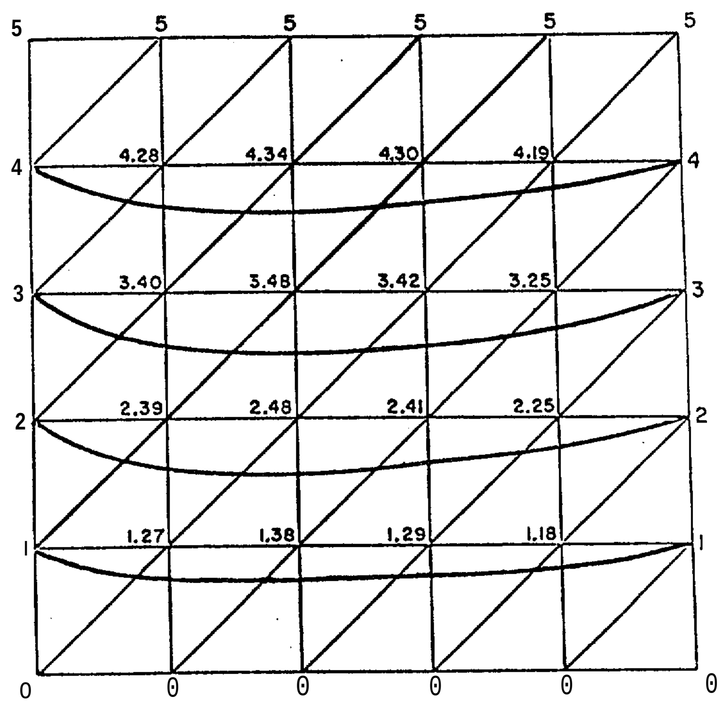


Figure 12.

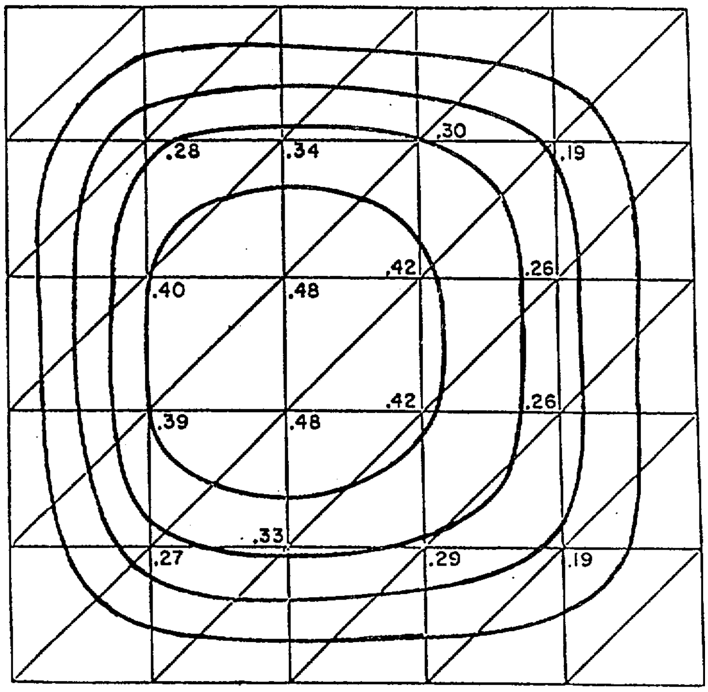


Figure 13.

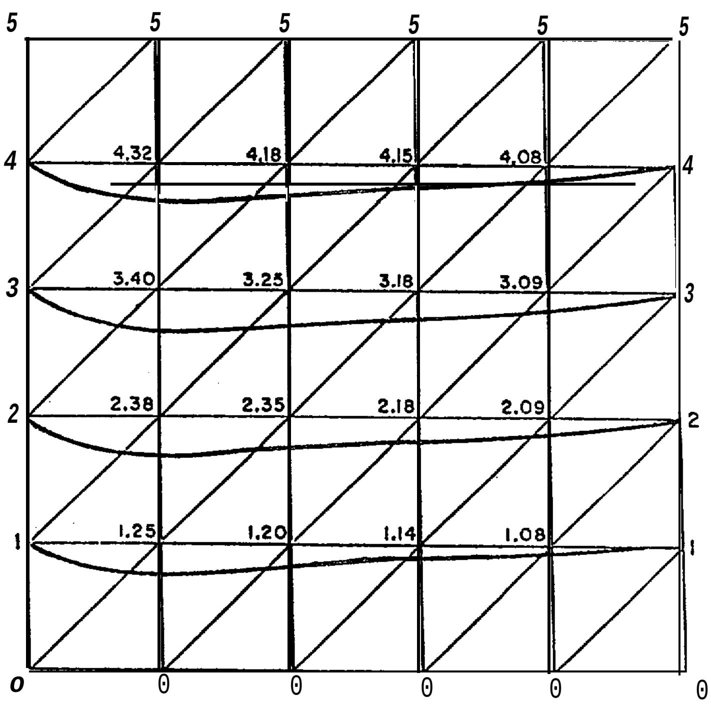


Figure 14.

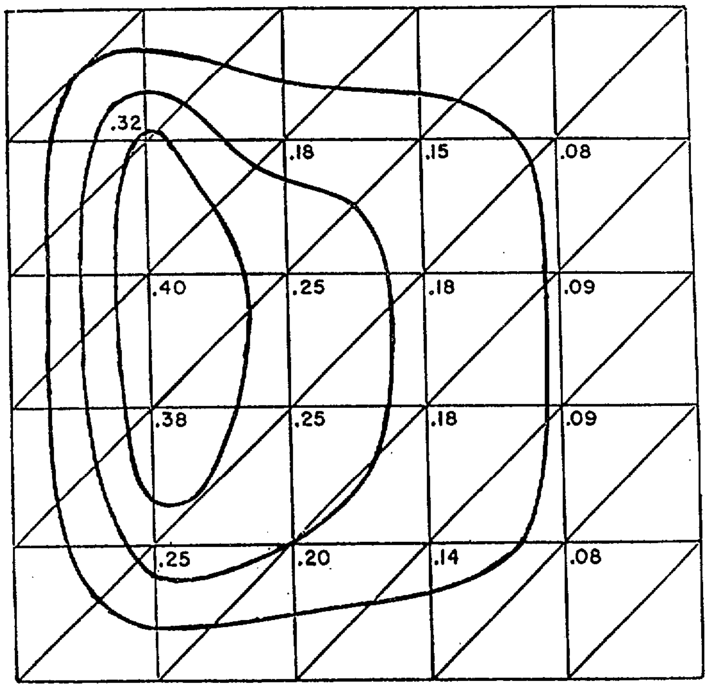


Figure 15.

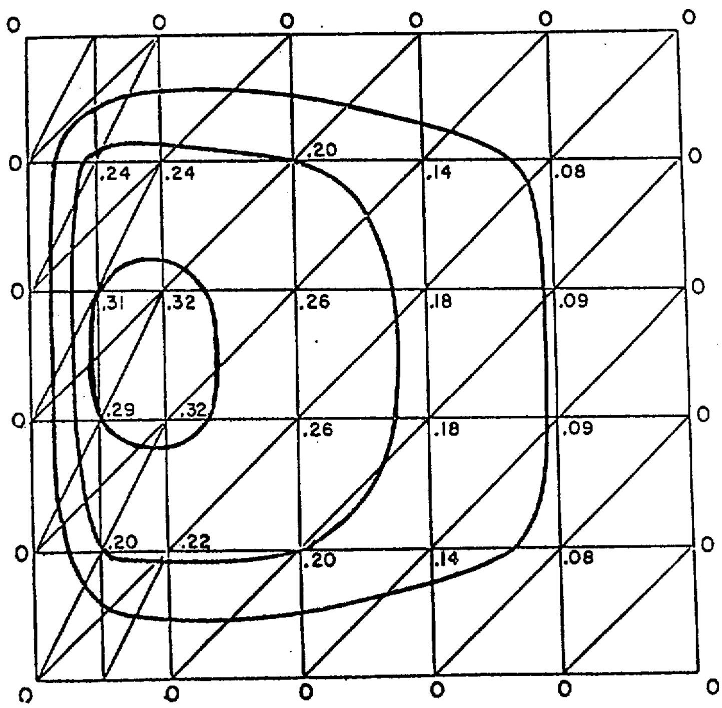


Figure 16.

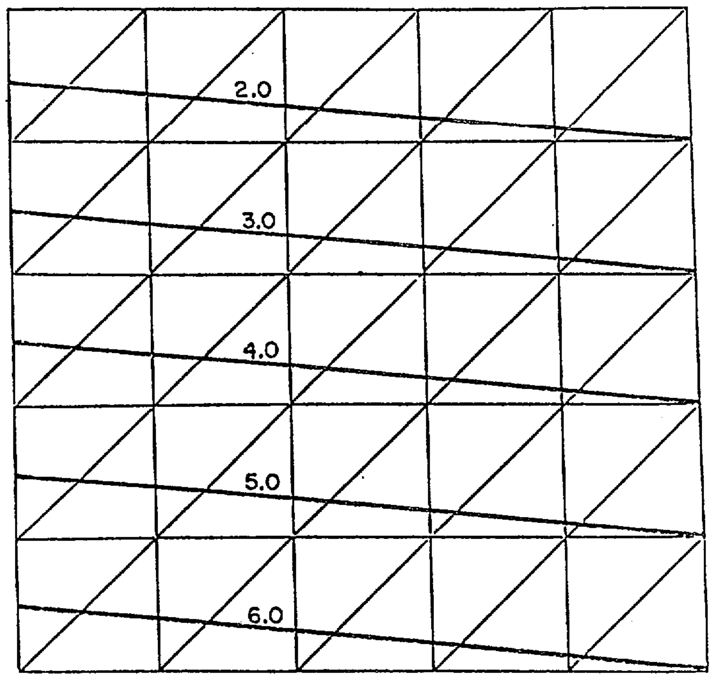


Figure 17.

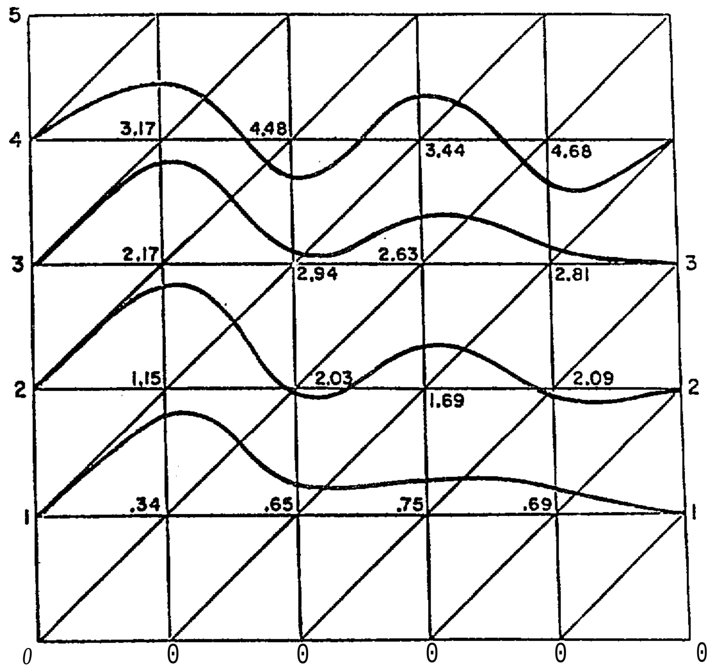


Figure 18.

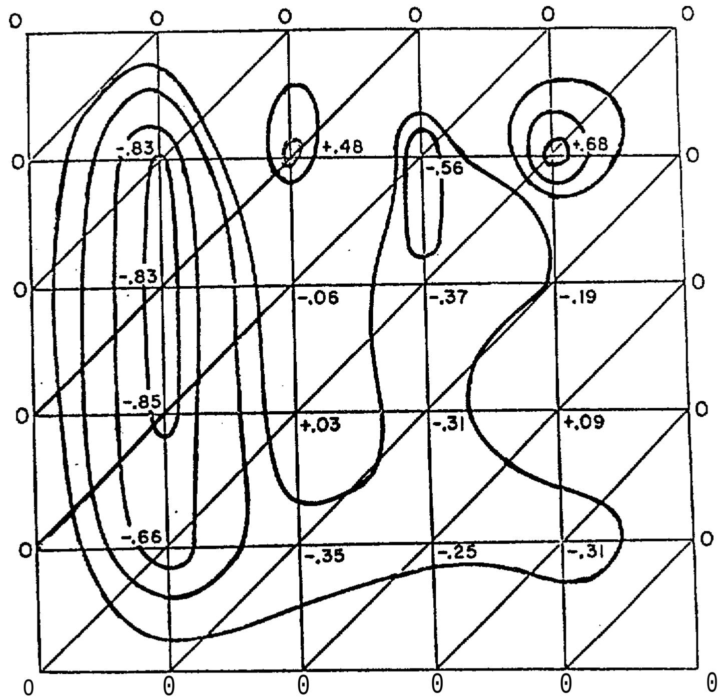


Figure 19.

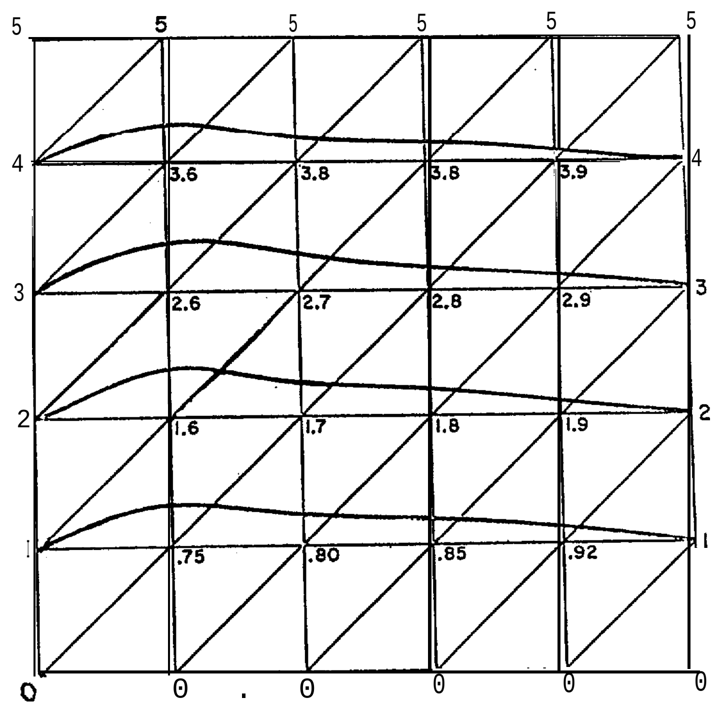


Figure 20.

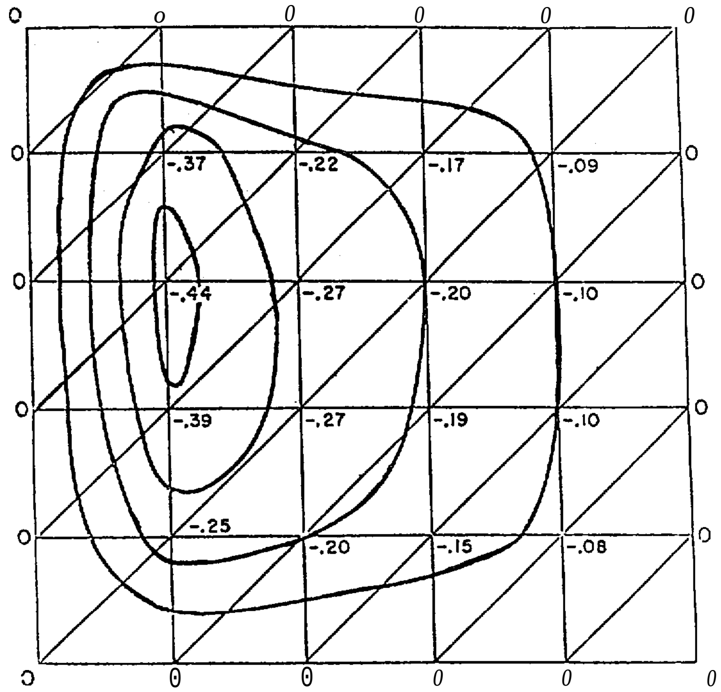


Figure 21.

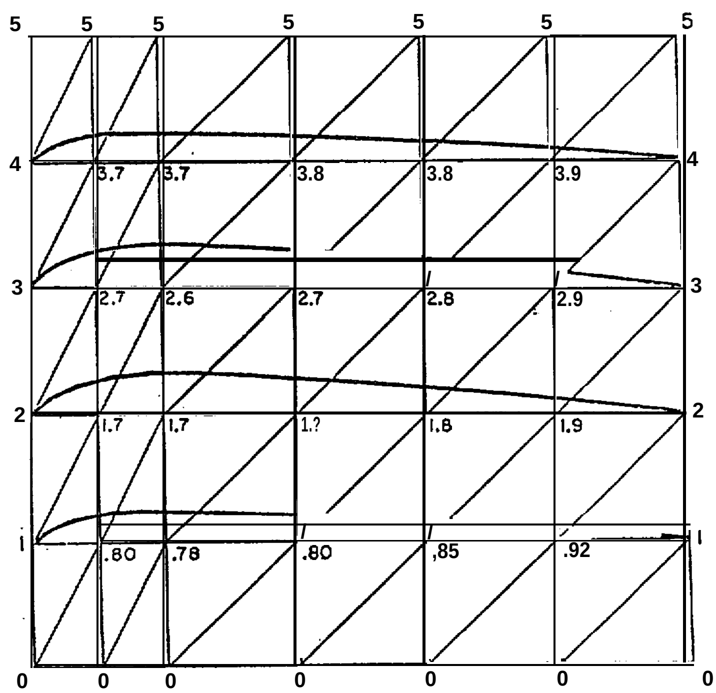
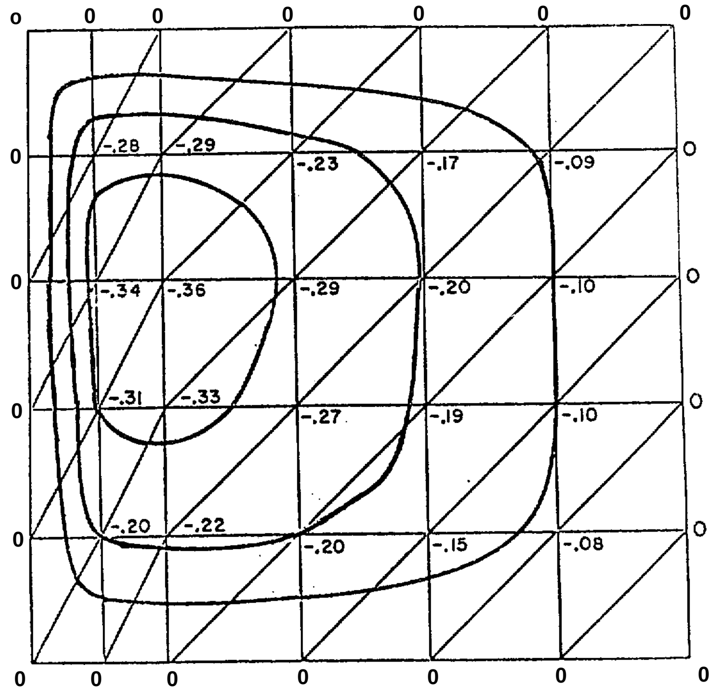


Figure 22.



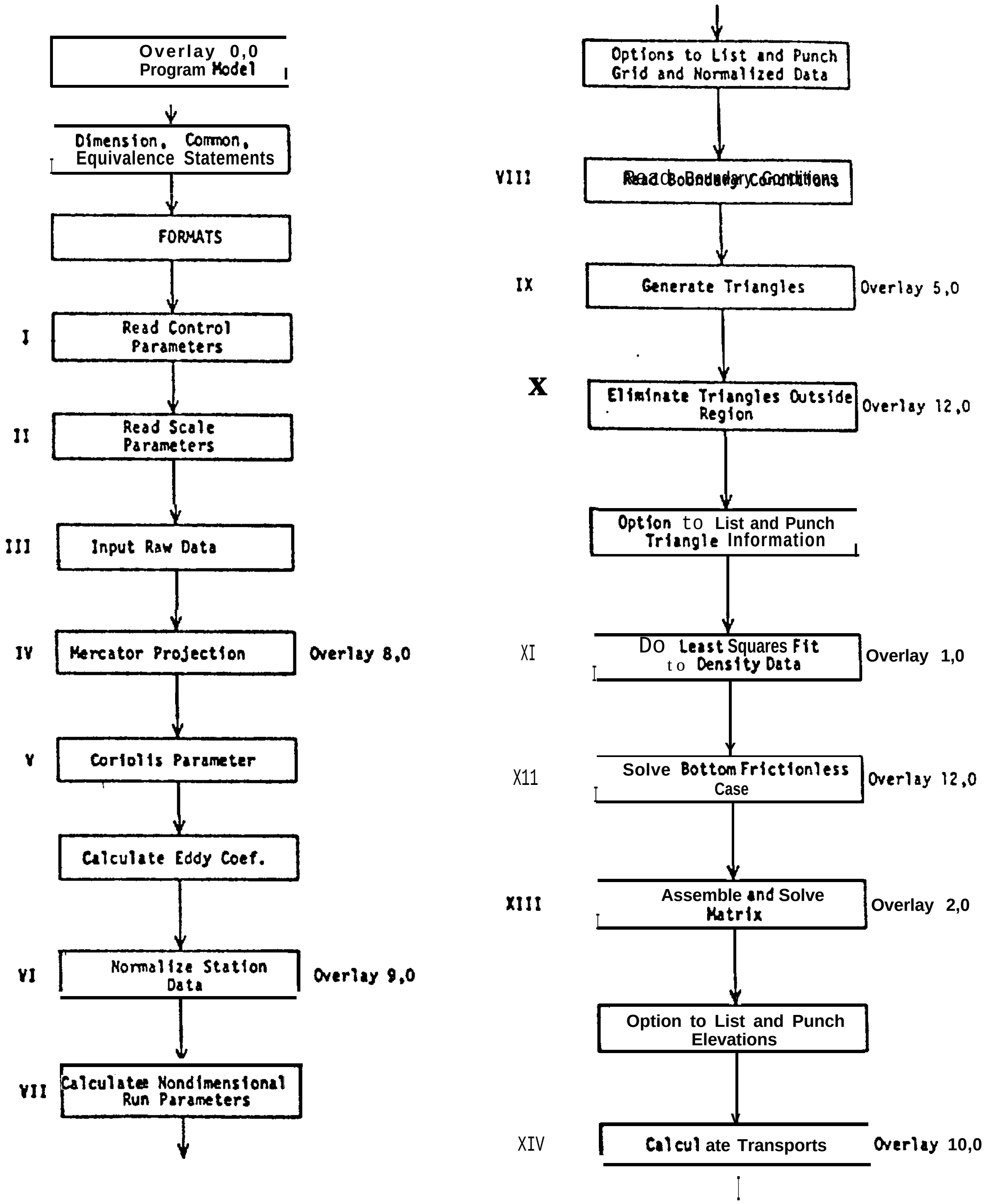
0
0
0
0
0
0
0
0
-.28
-.29
-.23
-.17
-.09
0
0
-.34
-.36
-.29
-.20
-.10
0
0
-.31
-.33
-.27
-.19
-.10
0
0
-.20
-.22
-.20
-.15
-.08
0
0
0
0
0
0
0
0

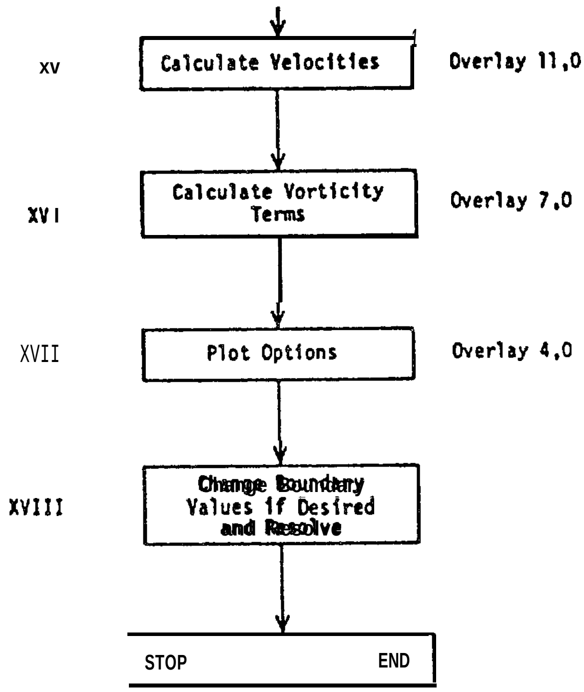
Figure 23.

Appendix 1

REVISED FLOW CHART OF

DIAGNOSTIC MODEL




XV
Calculate Velocities
Overlay 11,0
XVI
Calculate Vorticity Terms
Overlay 7,0
XVII
Plot Options
Overlay 4,0
XVIII
Change Boundary Values if Desired and Resolve
STOP
END

APPENDIX II LISTING

```
KRONOS L419.39.        77/09/13.
      OPERATING SYSTEM
    JOB ORIGIN = BATCH.
USER NUMBER/ID =  GW
   JOBCARD NAME = GLENS00
```

```
NN         hN  EEEEEEEEEEEE     111        YY         YY  KK          KK
NNN        NN  EEEEEEEEEEEE    1111        YY         YY  KK         KK
NNNN       NN  EE              1 11        YY         YY  KK        KK
NN NN      NN  EE                11        YY         YY  KK       KK
NN  NN     NN  EE                11        YY         YY  KK      KK
NN   NN    NN  EE                11        YY         YY  KK     KK
NN    NN   NN  EE                11        YYY       YYY  KK   KK
NN     NN NN   EEEEEEEE          11         YYY     YYY   KK KK
NN      NNNN   EEEEEEEE          11          YYY  YYY     KKKKK
NN       NNN   EE                11            YYYY       KKKKKK
NN        NN   EE                11             YY        KK   KK
NN        NN   EE                11             YY        KK    KK
NN        NN   EE                11             YY        KK     KK
NN        NN   EE                11             YY        KU      KK
NN        NN   EEEEEEEEEEEE  111111111111       YY        KK       KK
NN        NN   EEEEEEEEEEEE  111111111111       YY        KK        KK
```

```
******************************

OUTPUT FOR:  CLIFF FRIDLIND

             PACIFIC MARINE ENVIRONMENTAL LABORATIORY

             3711 15TH AVE. N. E.

             SEATTLE, WASHINGTON  98105

FTS PHONE: 399-4850

******************************

● W*W**4******W4 ● *+m*4*4?44*+**
```

```
      OVERLAY(MODEL,0,0)
      PROCRAM MODEL(FILEI,OUTPUT,FILEO,FILE1,TAPE5=FILEI,TAPE6=OUTPUT,TA
     CPE7=FILEO,TAPE1=FILE1,TAPE99)
C
C        QUASIGEOSTROPHICMODELWITH ANEKMANBOTTOM   FRICTION   LAYER
C     THE MAXIMUM NUMBER CF STATIONS THIS PROGRAM IS SETUP TO TAKE
C     IS 200. IF MORESTATICNS ARE ANALYSED,THE DIMENSIONSTATEMENTS
C     MUST BEINCREASED.  THE PROGRAM AS IT  STANDS TAKESABOUT750COCM
C     OCTAL IF THE U.W.GRAFIX LIBRARY IS ATTACHED.
C
C
      DIHENS1ON XS(100),YS(100)
      DIMENSION ALAT(200),ALONG(200)
      DIMENSION LAT(200),LONG(200)
      COMMON/LSCOEF/SA(4),SD(4),STNDVA,STNDVD
C
      COMMON/IALPHA/ALPHA(200)
      COMMON/IDELTA/DELTA(200)
      COMMON/IN/N(300)
      COMMON/BOUND/IB(75),BV(75),NBV
      COMMON/IA/A(3,3)
      COMMON/EXTRA/EX(300)
      COMMCN/IB/B(3)
      COMMCN/CALC CMP/XXMAX,YYMAX,ISTART,NBC
      COMMCN/IALP/ALPH(3)
      COMMON/ICONST/CONST1,CONST2
      COMMON/IDEPTH/DEPTH(200)
      COMMON/IGRAD/DALPHAX,DALPHAY,DDEPTHX,DDEPTHY,DEX,DEY,AREA
      COMMON/IHEIGHT/HEIGHT(200)
      COMMON/IIP/IP(3,350)
      COMMON/INT/INTP(4500)
      COMMON/IRHS/RHS(200)
      COMMON/SCALES/USCALE,DSCALE,ALSCALE,G,E,Q,GAMMA,FO,EDDY
      COMMON/WIND/TAUX,TAUY,CURL
      COMMON/IWORK/VALP(4500)
      COMMON/IX/X(200)
      COMMON/IY/Y(200)
      COMMON/NUMB/NVRTX,NTRI,LIST,IPUNCH,NEWBV
      COMMON/CUTOFF/NIBP,NFLX,NOMAT,JJ2,DEEP
C
      COMMON/XYPLOT/XPLOT,YPLOT
      COMMON/GDIV/X1(102),Y1( 102), X2(102), Y2(102),NX,NY,XDIV,YDIV
      COMMON/DEC/NDEC
      COMMON/DEG/DEGREE
      COMMON/CHSIZE/SIZE
      COMMON/BLCK/IBLOCK
      COMMON/OBJ/XRANGE,YRANGE,XMIN,YMIN,YBT,SLFT,YTP,XRGHT
      COMMON/SUB/XLEFT,YBOT,XSCALE,YSCALE
      COMMON/STNCH/NCHR
      COMMON/LTYPE/NLINE
      COMMON/NPT/NPTS
      EQUIVALENCE (HEIGHT(1),LONG(1))
      EQUIVALENCE(ALAT(1),Y(1))
```

```
      EQUIVALENCE(ALONG(1),X(1))
      EQUIVALENCE(IP(1,1),LAT(1))
C
C
C     ***** ***** **** WARNI NG.MIXEDMODE OPERATIONS ABOUND*************
C
C
C      WE  BEGIN  WITH ALL THE FORMATS WENEED. NOTE THAT THE FORMAT
C     NUMBERS ARE ALL MULTIPLES OF 5.
C
    5 FORMAT(15I5)
   10 FORMAT(I3,I3,F6.2,I5,F6.2,F10.4,F14.4,F7.1,F6.3,F5.3,F6.3)
   15 FORMAT(*1*)
   20 FORMAT(*0*,*STAT.*,5X,*LATITUDE*,5X,*LONGITUDE*,6X,*ALPHA*,11X,*DE
     *LTA*,10X,*DEPTH*)
   25 FORMAT(* ● t14t5XS13tIX,F4.IS5X ,I3,1X,F5.2,4X,F9.4,6X,F13.4,5X,F7.1
     ≠,6X,F4.2,5X,F4.2,6X,F10.7)
   30 FORMAT(*0*,*TRIANGLENO.*,10X,*VERTEX 1*,10X,*VERTEX 2*s10X,"VERTE
     ●X 3*)
   35  FORMAT(**,4X,I3,19X,I3,14X,I3,15X, 13)
   40  FORMAT(*0*,*TRIANGLE NO.*,35X,*VERTICES*)
   45 FORMAT(* ● g4Xs13S10XS*{*sF50  2,*,*,F5.2,*)*,10X, ● [*tF502t*s*sF5,2,  *
     *l**  1OX,*(**FS*2***4*F5*  2,*)*)
   50 FORMAT(I5,F20.3)
   55 FORMAT(*0*,*BOUNDARY VAL. NO,*,1OX,*GLOBAL LABEL*,10X,*BOUNDARY VA
     ALUE*,9X,*CENTIMETERS*)
   60 FORMAT(* *  ,8X,I3,20X,I3,18X,F7.3,15X,F8.3)
   65  FORMAT(*0*,*PROGRAMTERMINATED,TRIANGLE*,2X,I3,2X,*HAS BOUNDARY P
     ≠OINTS*,2X,I3,*,*,2X,I3,*,*,2X,*AND*,2X,I3,2X,*FOR VERTICES*)
   70 FORMAT(1H0,*PROGRAM STOPPED TO CHECK NEW BOUNDARY CONDITIONS*)
   75 FORMAT(*0*,5X,I3,14X,I3, I1O*3OX,I3)
   80 FORMAT(*0*,*BOUNDARY VALUES*,/,*0*,*B.P.*,6X,*LATITUDE* ,10X,*LONGI
     ≠TUDE*,10X,*VALUE*)
   85 FORMAT(* *,I3,7X,I3,1X,F4.1,10X,I3,1X,F4.1,11X,F5.2)
   90 FORMAT(*0*)
   95 FORMAT(*0*,*STATNO.*,10X,*GLOBAL NO.**IOX,"X COOR.  *,IOX,"Y-COOQ'  )
  100 FORMAT(* *,2X,I3,15X,I3,13X,F6.2,10X,F6.2)
  105 FORMAT(*1*,*RECHECK BOUNDARY TRIANGLE*,I3)
  110 FORMAT(*1*,*STATION*,2X,I4,2X,*NOT ONE OF LISTED STATIONS")
  115 FORMAT(*0*,*GLOBAL LABEL*,10X,*X-COOR*,10X,*Y-COOR*,17X,*ELEVATION
     H NONDIM.*,5X,*ELEVATIONS IN CM.*,/)
  120 FORMAT(* ● q4XS13S15XSF6.2s10X,F6.2t  20X,F9.5,12X,F10.5)
  125 FORMAT(1H0,*RADIUS USED IN  MERCATOR PROJECTION IS*,F10.5)
  130 FORMAT(*1*,*BOUNDARYPOINTS*,1X,I3,1X,*AND ● t$.Xt13tlXS*ARE IDENTICA
     ≠L*)
  135  FORMAT(**)
  140 FORMAT(7F10.2)
  145 FORMAT(**,*THISLISTINGISBEFORE WE ELIMINATE TRIANGLES OUTSIDE
     #OF CUR DOMAIN*)
  150 FORMAT(7F10.3)
  155 FORMAT(1H0,*X=0 LINE IS*,F7.4,1X,*RADIANS OF LONGITUDE WEST*)
  160 FORMAT(1H,*Y=0 LINE IS*,F7.4,1X,*RADIANS OF LATITUDE NORTH*)
  165 FORMAT(* ● j*STAT,*?9X,*GL. LAB.*,9X,*ALPHA*,11X,*DELTA*,11X,*DEPTH
```

```
     ≠*)
 170 FORMAT(1X,I3,12X,I3 ,9X,F9.5,6X,F11.5 ,7X,F9.5,10X,F9.4)
 175 FORMAT(* *,*NORMALIZED VALUES*,/)
 190 FORMAT(* '**Transport AND MEAN VELOCITIES NONDIMENSIONALIZED*,/)
 195 FORMAT(1X,2F10.2,50X,*STA.*,2X,I3)
 200 FORMAT(F10.4,F15.4,F10.2,F10.4,F11.4,15X,*STA.*,2X,I3)
 205 FORMAT(3I5,55X,*TRIA.*, 1X,I4)
 210 FORMAT(F10.4,60X,*HEIGHT*,1X,I3)
 215 FORMAT(* *,*SCALE PARAMETERS*)
 220 FORMAT( *0*,*VELOCITY*,7X,F6. 3,1X,*METERS/SEC*,//,* *,*DEPTH*,10X,F
    ≠7.1,1X,*METERS*,//,**,*LENGTH*,9X,F8.1,1X,*METERS*,//,* ● ,$GRA'JIT
    ≠Y*,8X,F7.2,1X,*M/(SEC SQ)*,//,* *,*PERT. DENSITY*,2X,F6.4,1X,*GM/(
    ≠CM CU)*,//,* *,*CONST. DENSITY*,1X,F7.3,1X ,*GM/(CM CU)*,//,* *,*GA
    ≠MMA*,10X,F7.1,1X,*GM/ (CM SQ)*,//,* *,*CORIOLIS*,7X,F9.7, 1X,*1/SEC*
    ≠)
 225  FORMAT(*0*,*NONDIMENSIONAL RUN PARAMETERS*,//,* *,*CONST1=GEH/QFUL
    ≠=*,F5.2,//,* *,*CONST2=GAMMA/QH=* ,F7.5)
 230 FORMAT(**,*VELOCITY IS IN CENTIMETERS pER SECOND*,/)
 235  FORMAT(**,*ELEVATICN SCALE FACTOR IS*,F8.3,1X,*CM.*)
 245 FORMAT(3F15.8)
 250 FORMAT(F10.4,F15.4,F10.4,36X,*STA.*,1X,I3)
 255 FORMAT(*MEAN CORICLIS IS*SF1OO8)
 260 FORMAT(*WIND STRESS AND CURL VALUES*,3F10.4)
 265  FORMAT(* *,*RAW STATION DATA*,/, ● *,*TAUX=*,F10.4,5X,*TAUY=*,F10.
    ≠4,5X,*CURL=*,F10.4)
 270 FORMAT(*ALPHA COEF.*,5F10.5)
 275 FORMAT(*DELTA COEF.*,5F10.5)
 280 FORMAT(*0*,*EDDY COEFFICEINT=*,F10.3,1X,*GM/ (CM SEC)*)
 285 FORMAT(1H0,10H**********,2X,*GLOBAL MATRIX IS SINGULAR, PROBLEM
    CTERMINATED*,10H**********)
C
C
C     HERE WE READ IN CONTRCL VALUES WHICH WILL TELL US WHAT WE wANT OUT
C     AND WHAT WE WANT DONE
C
C     IF ANY OF THE THE PARAMETERS IS SET TO 1, THE PROGRAM WILL EXECUTE
C     THAT OPTION.  IF IT IS 09 THEN THE OPTION WILL BE BYPASSED
C     ALL LIST OPTIONS BEGIN WITH L* AND ALL PUNCH OPTIONS BEGIN WITH IP.
C
C     LRWDATA=1 MEANS TH4T THE RAW DATA WILL BE LISTED.                    DATA 1
C
C     NCGRID=1 MEANS THAT THE CARTESIAN GRID WILL BE GENERATED,            DATA 2
C          NOTE THAT IF THIS IS 09 THEN THE GRID IS READ IN.
C     LGRID=1 MEANS THAT THE CARTESIAN GRID WILL BE LISTED.
C     IPGRID=1 MEANS THAT THE CARTESIAN GRID WILL BE PUNCHED.
C
C     NORMAL=1 MEANS THAT THE RAW DATA WILL BE NORMALIZED,                 DATA 3
C          ONCE AGAIN IF THIS IS SET TO 0, THEN THE DATA WILL HAVE TO BE
C          READ IN.
C     LNORM=1 MEANS THAT THE NORMALIZED DATA WILL BE LISTED.
C     IPNORM=1 MEANS THAT THE NORMALIZED DATA WILL BE PUNCHED.
C
C     MAKETRI=1 MEANS THAT THE TRIANGLES WILL GE GENERATED.                DATA 3
```

```
C        IF THIS IS SET TO 0, THEN THE TRIANGLES HILL HAVE TO BE
C        READ INO
C     LTRI=1 MEANS THAT THE TRIANGLES WILL BE LISTED.
C     IPTRI=1 MEANS THAT THE TRIANGLE LABELS WILL BE PUNCHED.
C     ISTOP=1 WILL SEND THE PROGRAM TO THE GRAFIX ROUTINES AFTER
C        GENERATING THE TRIANGLES.  THIS ALLOHS THE MESH TO BE CHECKED
C        BEFORE PROCEEDING ON.
C
C     NOCORY=1 MEANS THAT THE CORIOLIS PARAMETER MILL BE GENERATED.  IF    OATA
C        THIS IS 0, THEN THE OATA HAS To BE READ IN.  HENCE, NOCORY=1
C        HILL ALSO PUNCH THE MEAN CORIOLIS VALUE.
C
C     NOCONST=1 MEANS THAT THE NONDIMENSIONAL RUN PARAMETERS HILL BE       DATA
C        GENERATED INSTEAD OF READ IN,
C     IPCONST=1 MEANS THAT THE NONDIMENSIONAL RUN PARAMETERS HILL BE
C        PUNCHED,
C
C     LBV=1 HILL LIST THE BOUNDARY CONDITIONS THAT ARE READ IN.            DATA
C
C     NOELEV=I MEANS THAT THE PROGRAM WILL GENERATE THE SURFACE            DATA .
C        ELEVATIONS.  THE ALTERNATIVE IS TO REAO THEM INo
C     LELEV=1 MEANS THAT THE PROGRAM WILL LIST THE SURFACE ELEVATIONS.
C     IPELEV=1 MEANS THAT THE PROGRAM WILL PUNCH THE SURFACE ELEVATIONS.
C
C     NOTRANS=1 MEANS THAT THE PROGRAM HILL GENERATE THE TRANSPORTSO       OATA
C     IPTRANS=1 MEANS THAT THE PROGRAM HILL PUNCH THE TRANSPORT INFORMA.
C
C     NOVELC=1 MEANS THAT THE VELOCITIES HILL BE GENERATED.                OATA 1
C     IPVELO=1 MEANS THAT THE VELOCITY INFORMATION WILL BE PUNCHED.
C
C     NOTERM=1 MEANS THAT THE VORTICITY TERMS HILL BE LISTED.              DATA 11
C
C     NBC=NUMBER OF DIFFERENT BOUNDARY CONDITION SETS YOU HAVE.            DATA 12
C
C     NOPLOT=1 MEANS THAT THE PROGRAM HILL CO CALCOMP PLOTTING.            OATA 13
C     IWHAT=-1 MEANS THAT OhLY THE TRIANGLES HILL BE PLOTTED.
C     IWHAT=0 MEANS THAT BOTH THE TRIANGLES HILL BE PLOTTED AND THE
C        SURFACE Elevations WILL BE CONTOURED.
C     IWHAT=1 MEANS THAT CNLY THE SURFACE ELEVATIONS WILL BE CONTOURED.
C
C     LSF=1 MEANS THAT THE LEAST SQUARES FIT TO THE ALPHA ANO DELTA        DATA 14
C        FIELDS WILL BE DCNE.  IF THIS IS ZERO, THEN THE COEFFICIENTS
C        WILL HAVE TO BE READ IN.  CONSEQUENTLY, IF LSF=1 THE
C        COEFFICEINTS WILL AUTOMATICALLY BE PUNCHED.
C     LCOEF=1 MEANS THAT THE LEAST SQUARES FIT INFORMATION WILL BE
C        LISTED.
C
C     SMTHA=THE NUMBER OF STANDARD OBVIATIONS YOU WANT THE ALPHA FIELD     OATA 15
C        SMOOTHED TO. IF SMTHA=0, THE WATER IS MADE HoMOGENEOUS.  IF
C        SMTHA=-1., THEN NO SMOOTHING IS DONE.
C     SMTHD=DELTA SMOOTHING PARAMETER.  THE OPTIONS ARE IDENTICAL TO
C        THE ALPHA SMOOTHING OPTIONS.  SEE ABOVE FOR DETAILS.
C
```

```
C     IFILE=-1MEANSTHAT THE PROGRAM WILL PUT THE DECOMPOSED MATRIX        DATA 1
C          ONTO FILE1.
C     IFILE=0MEANS THAT THE PROGRAM HILL DECOMPOSETHEMATRIX BUT NOT
C          PUT IT ONTOFILE.
C     IFILE=1MEANSTHAT THE PROGRAMHILL READ THE DECOMPOSED MATRIX
C          FRCH FILE1.
C     *****REMEMBER TO REQUEST FILE1 AND CATALOG IT IF THE MATRIX IS
C          PLACEO ONTO FILE1. ALSO REMEMBER TO ATTACH THEFILE IF THE
C          THE MATRIX IS TO BE READ FROM THE FILE.*****
'C
C     NIBF=NUMBER OF INTERICR(ISLAND)BOUNDARY POINTS YOU HAVE.  ONE       DATA 1
C          ISLAND IS PERMITTED.
C     NFLX=NUHBER OF ONSHCRE BOUNOARY POINTS YOU HAVE.  THESE ARE THE
C          BOUNDARY POINTS WHICH DEFINETHEMAINLANC COASTLINE.
C
C     NOBCK=1MEANS THAT THE PROGRAM HILL ALTER THE OPEN BOUNDARY          DATA 1
C          CONDITIONS BY SOLVING THEBOTTOM FRICTIONLESS CASE.
C     NOINTG=1MEANS THAT THE INTEGRATION ALONG DEPTH CONTOURS WILL BE
C          DONE. THE ALTERNATIVE (NOINTG=0), IS TO READ IN THE
C          ELEvATION CHANGES ALONG THE DEPTH CONTOURS.
C     IPINTG=1MEANS THAT THE ELEVATION CHANGES ALONG THE DEPTH CONTOURS
C          HILL BE PUNCHED OUT.  NOTE THAT THE ELEVATION CHANGES ALONG
C          DEPTHCONTCURSARE FUNCTIONS ONLY OF THE DEPTH, ANO FORCING
C          FUNCTIONS, NCT OF THE BOUNDARY VALUES THEMSELVES.
C     IPBV=1 MEANS THAT THE NEWLYADJUSTEDBOUNDARY VALUES MILL BE
C          PUNCHED.
C     NOHALT=1MEANSTHAT THE PROGRAII HILL BE TERMINATE AFTER THE
C          BOUNDARY VALUES HAVE BEEN AOJUSTEO TO SEE IF THE BOUNDARY
C          CONDITIONS ARE REASONABLE.
C
C     DEEP=CUTOFFDEPTH AT WHICH THE BOUNOARY VALUES ARE OBTAINED FROM     OATA 1
C          DYNAMIC HEIGHT CALCULATIONS,  THESE BOUNDARY VALUES HILL NOT
C          BE ALTERED.
C          THE QUANTITY OEEP IS READ IN NEGATIVE METERS.
C
C
      READ(5,5)LRWDATA                                                    DATA
      READ(5,5)NOGRID,LGRID,IPGRID                                        DATA
      READ(5,5)NCRMAL,LNORM,IPNORM                                        DATA
      READ(5,5)MAKETRI,LTRI,IPTRI, ISTO?                                  DATA
      READ(5,5)NOCORY                                                     DATA
      REAO(5, 5)NOCONST,IPCONST                                           DATA
      READ(5,5)LBV                                                        DATA
      READ(5,5)NOELEV,LELEV,IPELEV                                        DATA
      READ(5,5)NOTRANS,IPTRANS                                            DATA
      READ(5,5)NOVELO,IPVELO                                              DATA 1
      READ(5,5)NOTERM                                                     DATA 1
      READ(5,5)NBC                                                        DATA 1
      READ(5,5)NOPLOT,IWHAT                                               DATA 1
      READ(5,5)LSF,LCOEF                                                  DATA 1
      READ(5,140)SMTHA,SMTHD                                              DATA 1
      READ(5,5)IFILE                                                      DATA 1
      READ (5,5)NIBP,NFLX                                                 DATA 1
```

```
      READ(5,5)NOBCK,NOINTG,IPINTG,IPBV,NOHALT                          DATA 1
      READ(5,150)DEEP                                                   DATA 1
C
      NEWBV=1
      ISTART=0
C
C     WE NOW PROCEEC TO READ IN THE SCALE PARAMETERS TO BE USEDFOR OUR
C     NONCIMENSIONALIZEDGRID AND OTHER SCALING.
C     THEVELOCITYSCALE,≠USCALE≠, IS IN METERS/SEC. THE DEPTH SCALE,
C     ≠DSCALE, IS IN METERS. THE HORIZONTAL LENGTH SCALE,≠ALSCALE≠,
C     ALSO IS IN METERS.
C     G, GRAVITY, IS IN METERS PER SECOND SQUARED
C     E, THE PERTURBATICN DENSITY, IS IN GM. PER CM.CUBED
C     Q, THE CONSTANTDENSITY IS ALSO IN GM. PER CM. CUBED,
C     GAMMA, THE BOTTOM FRICTION COEFFICIENT, IS INGM. PER CM.SQUARED
C     EODY IS THE EOOY COEFFICIENT CALCULATE FROMGAMMA.
C
      READ(5,150)USCALE,DSCALE,ALSCALE,G,E,Q,GAMMA
      DEEP=OEEP/OSCALE
C
C     NOW THE STATICN OATA IS REAO IN.
C     ALPHA IS ENTEREO IN (GM/CM**3)*METERS ANO DELTA IS ALPHA*METER
C     TAU, HINO STRESS, IS IN DYNES/(CM*CM)
C     CU?L IS REAO IN ASCYNES PER CM.CUBED.
C
C     AFTER THE LAST STATICNDATA CARD IS READ, A CARDWITH STATION
C     NUMBER ZERO IS READIN TO INDICATE THAT NO MORE STATION OATA
C     WILL BE INPUTED.
C
C     NOTE***THECEPTH,ALPHA,ANDDELTA VALUES ARE POSITIVE QUANTIIES
C
C
      IF(NOGRID.EQ.0.AND.NORMAL.EQ.0) GO TO 111
      I=1
    2 READ(5,10)N(I) ,LAT(I),ALAT (I),LONG( I),ALONG(I),ALPHA (I),DELTA(I),D
     ≠EPTH(I)
      DEPTH(I)=-DEPTH(I)
      IF(N(I).EQ.0) GO TO 4
      I=I+1
      GO TO 2
C
    4 NVRTX=I-1
      READ(5,245)TAUX,TAUY,CURL
C
C     WE EXIT FROMHEREWITH THE VALUE OF NVRTX, THE NUMBER OF VERTICES
C
C     HERE THE STATIONOATA IS ECHOCHECKED IF REQUESTED
C
      IF(LRWDATA.EQ.0) GO TO 8
      WRITE(6,15)
      WRITE(6,265)TAUX,TAUY,CURL
      WRITE(6,20)
      WRITE(6,90)
```

```
      DO 6 J=1, NVRTX
    6 WRITE(6,25)N(J),LAT(J),ALAT(J),LONG(J),ALONG(J),ALPHA(J),DELTA(J),
     ≠DEPTH(J)
    8 CONTINUE
C
C
  111 CONTINUE
C
      IF(NOGRID.EQ.0) GO TO 12
      IPUNCH=IPGRID
      CALL CVERLAY(5HMODEL,8, 0,0)
      XMAX=DALPHAX
      YMAX=DALPHAY
      YMIN=DDEPTHX
      RADIUS=DDEPTHY
      XMIN=DEX
      GO TO 13
   12 CONTINUE
C
C     *****NOTE THAT IF THE GRIo Subroutine ISBYPASSED, ONE NEEDS TO
C     READ IN YMIN,YMAX,YYMAX, AND XXMAX
C
      READ(5,5)NVRTX
      DO 3001 I=1,NVRTX
      READ(5,195)X(I),Y(I),N(I)
 3001 CONTINUE
      READ(5,140)YMIN,YMAX,YYMAX,XXMAX
   13 CONTINUE
C
C
C     NOWWE GO ON TO CALCULATE F KNOT, THE MEAN CORIOLISVALUE.
C
      IF(NOCORY.EQ.0) GO TO 22
      CALL BETA1(YMIN,YMAX,FO)
      WRITE(7,255)FO
      GO TO 23
   22 READ(5,255)FO
   23 CONTINUE
C
      EDDY=GAMMA*GAMMA*2.*FC/Q
C
C     HERE THE STATION DATA IS NORMALIZED IF REQUESTED.
C
C
      IF(NORMAL.EQ.0) GO TO 24
      IPUNCH=IPNORM
      CALL CVERLAY(5HMODEL,9,0,0)
      GO TO 6026
C
   24 CONTINUE
      00 1012 I=1,NVRTX
      READ(5,250)ALFHA(I),DELTA(I),DEPTH(I),N(I)
 1012 CONTINUE
```

```
      READ/5,260)TAUX,TAUY,CURL
C
C
C
 6026 CONTINUE
C     HERE WE LIST THE DIMENSIONAL COEFFICIENTS
C
      WRITE(6,15)
      WRITE(6,215)
      WRITE(6,220)USCALE,DSCALE,ALSCALE,G,E,Q,GAMMA,FO
      WRITE(6,280)EDDY
C
C     HEREWEHAVETHEOPTICN TO GENERATE THE NON DIMENSIONAL RUN
C     PARAMETERS OR READ THEMIN
C     WEALSO HAVE THE, OPTION TO PUNCH THEM UP FOR FUTURE USE
C
      IF(NOCONST.EQ.1) GO TO 89
C
C     HERE NONDIMENSIONAL RUN PARAMETERS ARE READ IN
C     THESE ARE THE CONST COEFFICIENTS IN THE EQUATION
      READ(5,245)CONST1,CONST2
C
      GO TO 91
   89 CONTINUE
      CONST1=(G*E*DSCALE)/(Q*FO*USCALE*ALSCALE)
      CONST2=GAMMA/(Q*DSCALE*100.)
   91 CONTINUE
      WRITE(6,225)CONST1.CONST2
C
C     THIS IS AN OPTION TO OUTPUT GRIO OATA
C
      IF(LGRID.EQ.0) GO TO 16
      WRITE(6,15)
      WRITE(6,95)
      WRITE(6,90)
      DO 14 I=1,NVRTX
      HRXTE(6,1001N[II SI,X(I) ,Y(I)
   14 CONTINUE
      WRITE(6,125)RADIUS
      WRITE(6,155)XMIN
      WRITE(6,160)YMIN
   16 CONTINUE
C
C
C     THIS ISAN OPTION YC LIST THENORMALIZED DATA
C
      IF(LNORM.EQ.0) GO TO 29
      WRITE(6,15)
      WRITE(6,175)
      WRITE(6,165)
      00 27 I=1,NVRTX
      WRITE(6,170)N(I),I,ALPHA(I),DELTA(I),DEPTH(I)
   27 CONTINUE
```

```
   29 CONTINUE
c
c     THIS IS AN OPTION TO PUNCH NORMALIZED DATA
C
      IF(IPCONST.EQ.0) GO TO 31
      WRITE(7,245)CCNST1,CONST2
   31 CONTINUE
C
C     ***********************************
C
c     WE NOW PROCEED TO READ IN THE TRIANGLES IF NECESSARY.
c     IF MAKETRI=0, THEN THE TRIANGLE VERTEX NUMBERS WILL BE READ IN
c     AND NO TRIANGLES WILL BE GENERATED.
C
      IF(MAKETRI.EQ.1) GO TO 902
c     HERE NTRI, THE NUMBER OF TRIANGLES ARE READ IN
  901 READ(5,5)NTRI
c     NOW TO READ IN THE GLOBAL LABELS OF EACH TRIANGLE VERTEX
      DO 903 I=1,NTRI
      READ(5,5)(IP(J,I),J=1,3)
  903 CONTINUE
  902 CONTINUE
c
c
C     WE NOW READ IN THE BOUNDARY VALUES AND THEIR STATION NUMBERS
C     SURFACE ELEVATIONS ALONG THE BOUNDARY SHOULD BE IN CM.
C
C
c
C     *****************
c     REMEMBER, IF AN ISLAND IS PRESENT, ITS BOUNDARY CONDITIONS SHOULD
C         BE READ IN FIRST IN CLOCKWISE ORDER. THEN THE INSHORE BOUNDA-
c         RY CONDITIONS SHOULD BE READ IN COUNTERCLOCKWISE ORDER.  THESE
C         ARE THEN FOLLOWED BY THE REST OF THE BOUNDARY CONDITIONS READ
C         IN COUNTERCLOCKWISE ORDER.
C     THE BOUNDARY Elevations FOR THE NO FLUX BOUNDARIES WILL BE
c           CALCULATE RELATIVE TO THE FIRST NO FLUX STATION. THEREFORE,
C           IF AN ISLAND IS PRESENT, THE ELEVATION OF THE FIRST ISLAND
c           BOUNDARY POINT NEEDS TO BE DEFINED.  THEN THE ELEVATION OF
c           THE FIRST ONSHORE, COASTLINE BOUNDARY STATION MUST ALSO BE
c           SPECIFIED.  ALL OTHER NO FLUX BOUNDARY ELEVATIONS DONT NEED
c           TO BE SPECIFIED.
c     A BLANK CARD SHOULD BE READ IN AFTER ALL THE BOUNDARY CONDIDTIONS
c     HAVE BEEN READ.
c     ● **+****4*+?**?****
c
      DIM=G/(FO*USCALE*ALSCALE*Q*100.)
      I=1
   58 READ(5,50) ISTAT,BVAL
      IF(ISTAT.EQ.0) GO TO 402
      DO 54 J=1,NVRTX
      IF(ISTAT.NE.N(J)) GO TO 54
      IB(I)=J
```

```
      BV(I)=BVAL*DIM
      I=I+1
      GO To 58
   54 CONTINUE
      HRITE(6,110)ISTAT
      GO TO 2001
  402 CONTINUE
   62 NBV=I-1
C
C     HERE, WE EXIT WITH, NBV, THE NUMBER OF BOUNDARY POINTS
C
C
C     HERE HE HAVE AN OPTION TO ECHO CHECK THE BOUNDARY CONDITIONS
C
      IF(LBV.EQ.0) GO TO 746
      WRITE(6,15)
      WRITE(6,55)
      WRITE(6,90)
      DO 64 I=1,NBV
      BVV=BV(I)/DIM
      WRITE(6,60)I,IB(I),BV(I),BVV
   64 CONTINUE
  746 CONTINUE
C
C     NOW GENERATE THE TRIANGLES IF REQUESTED.
C
      IF(MAKETRI.EQ.0)GO TO 904
      IPUNCH=NIBP
      CALL OVERLAY(5HMODEL,5,0,0)
C
  904 CONTINUE
C
  401 CONTINUE
      IF(MAKETRI.EQ.0) GO TO 754
C
C     IF TRIANGLES WERE INTERNALLY GENERATED, WE WILL PROCEED TO
C     ELIMINATE THE TRIANGLES EXTERIOR TO THE REGION OF INTEREST.
C
      LIST=LTRI
      CALL OVERLAY(5HMODEL,6, 0,0)
  754 CONTINUE
C
      IF(LTRI.EQ.0) GO TO 57
C     TRIANGLE NUMBERS ALONG WITH THE GLOBAL LABELS OF EACH VERTEX IS
C     LISTED AND ON THE NEXT PAGE, THE VERTEX COORDINATES ARE LISTED
C
      WRITE(6,15)
      WRITE(6,30)
      WRITE(6,90)
      DO 26 I=1,NTRI
   26 WRITE(6,35)I,(IP(J,I),J=1,3)
      WRITE(6,15)
      WRITE(6,40)
```

```
      WRITE(6,90)
      DO 28 I=1,NTRI
      II=IP(1,I)
      IL=IP(2,I)
      IM=IP(3,I)
   28 WRITE(6,45)    I,X(II),Y(II), X(IL),Y(I L),X(IM),Y(IM)
   57 CONTINUE
C
C     THIS IS THE OPTION TO PUNCH THE TRIANGLE OATA
C
      IF(IPTRI.EQ.0) GO TO 157
      WRITE(7,5)NTRI
      0 0 156 I=1,NTRI
      WRITE(7,205)IP(1,I),IP(2,I),IP(3,I),I
  156 CONTINUE
  157 CONTINUE
C
C
C
C     WE NOW FIT THE ALPHA ANO DELTA TO A THIRD ORDER POLYNOMIAL BY
C     DOING A LEAST LEAST SQUARES FIT TO THE ALPHA AND DELTA FIELDS.
C
      IPUNCH=LSF
     LIST=LCOEF
      SA(1)=SMTHA
      SD(1)=SMTHD
      CALL CVERLAY(5HMODEL, 1,0,0)
C
C
  132 CONTINUE
C     *********
C
C     NOW PROCEED TO ALTER THE BOUNDARY CONDITIONS BY SOLVING THE
C     INVISCID CASE ALONG BOUNDARY DEPTH CONTOURS.
C
C
      IF(NOBCK.EQ.0) GO TO 68
      LIST=NOINTG
      IPUNCH=IPINTG
      CALL CVERLAY(5HMODEL,12,0,0)
      IF(IPBV.EQ.0) GO TO 67
      DO 66 I=1,NBV
      BVV=BV(I)/DIM
      J=IB(I)
   66 WRITE(7,50)N(J),BVV
   67 CONTINUE
      IF(NOHALT.EQ.0) GO TO 68
      WRITE(6,70)
      GO TO 2001
   68 CONTINUE
C
C      IF THE LEAST SQUARES COEF. TO THE BAROCL.FIELD WAS GENERATED
C     THEN WE WILL SAVE THE VALUESO
```

```
      IF(LSF.EQ.0) GO TO 69
C     NOW PROCEED To PUNCH THE VALUES UP SO WE DO NOT HAVE TO
C     REGENERATE THEM AGAIN.
      WRITE(7,270)SA(1),SA(2),SA(3),SA(4),STNDVA
      WRITE(7,275)SD(1),SD(2),SD(3),SD(4),STNOVD
   69 CONTINUE
C
      IF(ISTOP.EQ.1) GO TO 2999
C     ● ********
C
C
  133 CONTINUE
C      WE NOW ASSEMBLE THE MATRIX
C
      NOMAT=0
      IF(NEWBV.GT.1) NOMAT=-1
      IF(NOELEV.EQ.1) GO TO 164
      DO 162 I=1,NVRTX
      READ(5,210)HEIGHT(I)
  162 CONTINUE
      GO TO 134
  164 CONTINUE
      LIST=IFILE
      CALL OVERLAY(5HMODEL,2, 0,0)
      IF(NVRTX.LT.0) 60 TO 4001
C
  134 CONTINUE
C
C     THIS IS AN OPTION TC LIST T H E EVEVATION OF THE VERTICIES.
C
      IF(LELEV.EQ.0) GO TO 178
      WRITE(6,15)
      ELEV=(FO*USCALE*ALSCALE/G)*100.
      WRITE(6,235)ELEV
      WRITE(6,115)
      DO 176 I=1,NVRTX
      DIM=HEIGHT(I)*ELEV
      WRITE(6,120)I,X(I),Y(I)● HEIGtIT(I),OIM
  176 CONTINUE
  178 CONTINUE
C
C     THIS IS THE OPTION TO PUNCH THE ELEVATION DATA
C
      IF(IPELEV.EQ.0) GO TO 159
      0 0 158 I=1,NVRTX
      HRITE(7?210) HEIGHT(IISI
  158 CONTINUE
  159 CONTINUE
C    '
C     HERE HE HAVE THE OPTICN TO CALCULATE THE TRANSPORT AT THE CENTROID
C     OF EACH TRIANGLE
C
      IF(NCTRANS.EQ.0) GO TC 184
```

```
      IPUNCH=IPTRANS
      CALL OVERLAY(5HMODEL,10,0,0)
C
  184 CONTINUE
C
C     OPTION TO CALCULATE ANO LIST  EKMAN VELOCITY, BAROTROPICVELOCITY
C     THEIRSUM, ANO BOTTOM VELOCITY
C
      IF(NOVELO.EQ.0) GO 70 186
      IPUNCH=IPVELO
      CALL OVERLAY(5HMODEL,11,0,0)
  186 CONTINUE
C
C     OPTION TO CALCULATE ANO LIST THEDYNAMIC BALANCE TERMS.
C
      IF(NOTERM.EQ.0) GO TO 188
      CALL OVERLAY(5HMODEL,7,0,0)
  188 CONTINUE
C     WE NOW HAVE THE OPTION OF PLOTTING THE TRIANGLES AND LABELING
C     THE VERTICIESWITH THEIR ELEVATIONS.
C     ONCE AGAIN THE UNIV.CF HASH.*S N.P.S.SYSTEM IS UTILIZED.
C
 2999 CONTINUE
      IF(NOPLOT.EQ.0.AND.NBC.EQ.1) GO TO 2001
      IF(NOPLOT.EQ.0)GO TO 1999
C
C     WE NOW PROCEED TOPLOT THE SURFACE ELEVATIONS.
      LIST=IWHAT
      CALL OVERLAY(5HMODEL,4, 0,0)
C
C
 1999 CONTINUE
      IF(NEWBV.GE.NBC) GO TO 2001
      CALL NEWBVAL
      NEWBV=NEWBV+1
      IF(JJ2.EQ.0) GO TO 133
      GO TO 132
C
C
C
C     IF THE FORCING FUNCTIONSAREALTERED, THEN THE NEWVALUES ARE
C     READ IN  SUBROUTINENEWBVAL ALONG WITH ANY NEWBOUNDARY VALUESO
C     THEN ONE REASSEMBLES THE RIGHT HANO SIOE  AND READJUSTS THE NEW
C     BOUNDARY VALUES BY SETTING JJ2 INCOMMON BLOCK CUTOFF TO 1 IN
C     SUBROUTINE NEWBVAL.  THIS HILL SENO THE PROGRAM INTO BCHK. IF
C     ONLY THE BOUNDARY VALUES ARE CHANGED IN SUBROUTINE NEWBVAL, THEN
C     JJ2=0 ANO ONLY THENEWBOUNDARY VALUES ARE ALTEREDAND NOT THE
C     ENTIRE RIGHT HAND SIDE.
C
C
 4001 WRITE(6,285)
 2001 STOP
      ENO
```

```
      OVERLAY(MODEL,1,0)
      PROGRAM LEAST
      COMMON/NUMB/NVRTX,NTRI,LIST,IPUNCH
      COMMON/IALPHA/ALPHA(200)
      COMMON/IDELTA/DELTA(200)
      COMMON/LSCOEF/SA(4),SD(4),STNDVA,STNDVD
      COMMON/IHEIGHT/HEIGHT(200)
      COMMON/IDEPTH/DEPTH(200)
  270 FORMAT(*ALPHA  CCEF**,5F1O,5)
  275 FORMAT(*ALPHA COEF.*,5F10.5)
      SMTHA=SA(1)
      SMTHD=SD(1)
      LSF=IPUNCH
      IF(LSF.EQ.0) GO TO 166
      LCOEF=LIST
      CALL LQFIT(ALFHA,SA,STNDVA,LCOEF,NVRTX)
      LCOEF=-LCOEF
      CALL LQFIT(DELTA,SD,STNDVD,LCOEF,NVRTX)
C     ****NOTE, THE COEFFICIENTS ARE STORED INTO SA, AND SD*   SA HOLDS
c     THE COEFFIENTS FOR ALPHA ANO SO FOR DELTA.
      GO TO 167
  166 CONTINUE
c     HERE WE READ IN THE ALPHA AND DELTA COEFFICIENTS IF THEY WERE
c     NOT GENERATED,
      READ(5,270) SA(1),SA(2),SA(3),SA(4),STNDVA
      READ(5,275) SD(1),SD(2),SD(3),SD(4),STNDVD
  16? CONTINUE
c
C     WE NOW HAVE THE OPTION OF SMOOTHING THE ALPHA AND DELTA FIELDS.
      IF(SMTHA.EQ.-1.) GO TO 168
      CALL SMOOTH(ALPHA,SA,SMTHA,STNDVA,NVRTX)
  168 CONTINUE
      IF(SHTHOOEQ.-1O} GO TO 169
      CALL SMOOTH(DELTA,SD,SMTHD,STNDVD,NVRTX)
  169 CONTINUE
      END
```

```
      OVERLAY(MODEL,2,C)
      PROGRAM MATASS
      COMMON/IN/N(300)
      COMMON/IRHS/RHS(200)
      COMMON/IWCRK/VALP(4500)
      COMMON/INT/INTP(4500)
      COMMON/LSCOEF/SA(4),SD(4)
      COMMON/IALPHA/ALPHA(200)
      COMMON/IDEPTH/DEPTH(200)
      COMMON/IIP/IP(3,350)
      COMMON/IA/A(3,3)
      COMMON/IB/B(3)
      COMMON/IY/Y(200)
      COMMON/IX/X(200)
      COMMON/NUMB/NVRTX,NTRI,IFILE,MASL,NEWBV
      COMMON/WIND/TAUX,TAUY,CURL
      COMMON/IALP/ALPH(3)
      COMMON/SCALES/USCALE,DSCALE,ALSCALE,G,E,Q,GAMMA,FO,EDDY
      COMMON/BOUND/IB(75),BV(75),NBV
      COMMON/IHEIGHT/HEIGHT(200)
      COMMON/ICCNST/CONST1,CONST2
      COMMCN/IGRAD/DALPHAX,DALPHAY,DDEPTHX,DDEPTHY,DEX,DEY,AREA
      COMMCN/CUTOFF/NIBP,NFLX,JJ1,JJ2
C     WE NCW PREPARE TCASSEMBLE THE MATRIX BY SETTING IT TO ZERO
C     ANDBY SETTING THE INTEGER BOOK KEEPING ARRAY TO ZERO.
C
      IF(JJ1.EQ.-1) GO TO 199
      IF(NEWBV.GT.1) GO TO 201
      CALL SETMAT(NVRTX,IFILE,MASL)
c
c
c
c     WE NOW PROCEED TO CALCULATE GRADIENTS ANO ASSEMBLE THE MATRIX
c
  199 CONTINUE
      CALL CVERLAY(5HMODEL,2,1,0)
c
  201 CONTINUE
c
      CALL CVERLAY(5HMODEL,2,2,0)
      END
```

```
      A(3,1)=X(M)
      A(3,2)=Y(M)
      A(1,3)=1.0
      A(2,3)=1.0
      A(3,3)=1.0
C
      CALL TRIAREA(A, AREA)
      AREA=ABS(AREA)
C
      CALL GRAD(DEPTH(J),DEPTH(L),DEPTH(M),DDEPTHX,DDEPTHY,CDEPTH)
C
      CALL ALPHX(K,DALPHAX,DALPHAY,ALPHA,SA,ALPH)
C
C
C
      DO 104 I=1,3
      DO 102 II=1,3
  102 B(II)=0.
      B(I)=1.
C
      CALL GRAD(B(1),B(2),B(3),DSHAPEX(I),DSHAPEY(I),CSHAPE(I))
C
C
  104 CONTINUE
C
      CALL MATRIX(DSHAPEX,DSHAPEY,K,NVRTX,NTRI,IFILE,MASL)
C
C
C
  128 K=K+1
      IF(K.GT.NTRI) GO TO 132
      GO TO 72
C
  132 CONTINUE
C
C     NOW PROCEED TO ADD ON THE CONTRIBUTIONS FROM THE NO FLUX BOUNDARY
C     CONDITIONS*
C
C
      CALL BASS(VALP,RHS)
C
C     HERE WE ADD THE PRESCRIBED BOUNDARY CONDITONS TO THE RIGHT HAND
C     SIDE.
C
      CALL BC(NVRTX,NBV,IB,BV,MASL)
C
      END
```

```
      OVERLAY(MODEL,2,2)
      PROGRAM SCLN
      COMMON/NUMB/NVRTX,NTRI,IFILE,MASL,NEWBV
      COMMON/INT/INTP(4500)
      COMMON/IHEIGHT/HEIGHT(200)
      COMMON/IHCRK/VALP(4500)
      COMMON/BOUND/IB(75),BV(75),NBV
      COMMON/IRHS/RHS(200)
  100 FORMAT(E12.5,020)
      NROW=NVRTX
      NCOL=NVRTX
      SMALL=.0000001
      IF(NEWBV.GT.1) GO TO i
      IF(IFILE)2,2,4
    2 CONTINUE
C
c     HERE THE MATRIX IS DECOMPOSED.
c
      CALL DCPK(NROW,NCOL,IS,IR,IF,SMALL,INTP,VALP,MASL)
      IF(IS.EQ.1) GO TO 10
      IF(IFILE.EQ.0) GO TO i
c
c     OPTION TO CREATE FILE
c
   11 "00 6 I=1,4500
      WRITE(1,100)VALP(I),INTP(I)
    6 CONTINUE
      GO TC1
C
C     OPTICN TO REAC FROM FILE
C
    4 00 a I=1,4500
      JK=I-1
      READ(1,100)VALP(I),INTP(I)
      GO TO 1
    8 CONTINUE
C
    1 CONTINUE
C
C     WENOWADD THE BOUNDARY CONDITIONS   IN.   NOTE THEY   MUST BEADDED
C     INTO  VALP AFTER THE MATRIX HAS BEEN DECOMPOSED.
      00 112 K=1,NRCW
      VALP(K)=RHS(K)
  112 CONTINUE
C
      CALL SLVK(NROW,NCOL,IE,INTP,VALP)
      IF(IE.EQ.1) GO TO 12
      GO TO 21
   10 WRITE(6,15)
   15 FORMAT(*0*,*SING IS 1, MATRIX IS SINGULAR*)
      NVRTX=-NVRTX
       GO TO 301
   12 WRITE(6,20)
   20 FORMAT(*0*,*ERROR IS 1, DIVISION BY ZERO IN SLVK*)
      NVRTX=-NVRTX
      GO TO 301
   21 CONTINUE
      00 22 I=1,NROW
      J=I+NROW
   22 HEIGHT(I)=VALP(J)
  301 CONTINUE
      ENO
```

```
      OVERLAY(MODEL,4,0)
      PROGRAM DRAW
      COMMON/IRHS/RHS(200)
      COMMON/BOUND/IB(75),BV(75),NBV
      COMMON/SCALES/USCALE,DSCALE,ALSCALE,G,E,Q,GAMMA,FO,EDDY
      COMMON/IX/XA(200)
      COMMON/IY/YA(200)
      COMMON/IIP/IP(3,350)
      COMMCN/NUMB/NVRTX,NTRI,IWHAT,IPUNCH,NEWBV
      COMMON/IN/N(300)
      COMMCN/IHEIGHT/HEIGHT(200)
      COMMON/CALCCMP/XXMAXA,YYMAXA,ISTART,NBC
      DIMENSICN CON(20)
      DIMENSION TLAEEL(3),X(200),Y(200)
c     THIS SECTION IS FORCALCOMPPLOTTIN
C     THE UNIVERSITY OF WASHINGTON/S NUMERICAL PLOTTIN SYSTEM IS USED
c     WE HAVE THE OPTION TO ORAH LABEL ANO CONTOUR EACH POINT AND TRIANGLE
      XXMAX=XXMAXA
      YYMAX=YYMAXA
      DIM=(FO*USCALE*ALSCALE)*100./G
C
      0 0 302 I=1,NVRTX
      X(I)=XA(I)
      Y(I)=YA(I)
  302 CONTINUE
      RATIO=XXMAX/YYMAX
      YSIZE=6.
      XSIZE=RATIO*YSIZE
      XINC=XSIZE+5.
      XINC=0.
      YSTART=.5
      XSTART=2.
      IF(IWHAT.EQ.1) GO TO 501
      ENCODE(30,34,TLABEL)
   34 FORMAT(27HTRIANGLES ANO GLOBAL LABELS)
      CALL SETUP(ISTART,1,XSIZE,YSIZE,TLABEL,1,0.,XXMAX,0.,YYMAX,05,05,X
     *START,YSTART)
      CALL DRTRI(X,Y,IP,NTRI,2,N)
      0 0 41 I=1,NVRTX
   41 RHS(I)=FLCAT(I)
      CALL ADVANC(0.,0.)
      CALL TRILABL(X,Y,IP,NTRI,.112,1)
      CALL ADVANC(0.,0.)
      CALL VRTXLB(X,Y,RHS,0,NVRTX,.098,1)
      IF(IWHAT)2,3,2
    3 CALL ADVANC(XINC,0.)
    2 CONTINUE
      ISTART=1
      IF(IWHAT.EQ.-1) GO TO 502
  501 CONTINUE
      ENCODE(30,33,TLABEL)
   3 3 FORMAT(26HSURFACE ELEVATION CONTOURS)
      CALL SETUP(ISTART,1,XSIZE,YSIZE,TLABEL,1,0.,XXMAX,0.,YYMAX,05, 05,X

     *START,YSTART)
      00 9 I=1,NVRTX
      HEIGHT (I)=HEIGHT(I)*DIM
    9 CONTINUE
      CALL KONTRI(X,Y,IP,CON,NTRI,NVRTX,8,HEIGHT,1)
      CALL FLTBND(X,Y,IB,NBV,1)
      CALL VRTXLB(X,Y,HEIGHT,1,NVRTX, .091,3)
  404 CONTINUE
      ISTART=1
  502 IF(NEWBV.LT.NBC) GO TO 1990
      CALL EXITPL
      GO TO 1
 1998 CALL ADVANC(XINC,0.)
    1 CCNTINUE
      END
```

```
      OVERLAY(MODEL,5,0)
      PROGRAM MESH
C
C     HERE WE GENERATE THE MESH.
C
      COMMON/IWORK/P(204,2),VERT(402,6)
      COMMON/INT/ISIDE(607,2),ITRI(403,3)
      COMMON/IX/X(200)
      COMMON/IY/Y(200)
      COMMON/IIP/IP(3,350)
      COMMON/NUMB/NVRTX,NTRI,LIST,NIBP,NEWBV
      COMMON/BOUND/IB(75),BV(75),NBV
      COMMON/CALCOMP/XXMAX,YYMAX,ISTART,NBC
C
      DO 101 I=1,NVRTX
      P(I,1)=X(I)
      P(I,2)=Y(I)
  101 CONTINUE
      CALL TRIAN(ISIDE,ITRI)
      DO 2 I=1,NTRI
      IS1=ITRI(I,1)
      IS2=ITRI(I,2)
      JP1=ISIDE(IS1,1)
      JP2=ISIDE(IS1,2)
      JP3=ISIDE(IS2,1)
      IF(JP1.EQ.JP3.OR.JP2.EQ.JP3) JP3=ISIDE(IS2,2)
      IP(1,I)=JP1
      IP(2,I)=JP2
      IP(3,I)=JP3
    2 CONTINUE
C
      END
```

```
      OVERLAY(MCDEL,6,0)
      PROGRAM OUTSIDE
      COMMON/BOUND/IB(75),BV(75),NBV
      COMMON/IIP/IP(3,350)
      COMMON/NUMH/NVRTX,NTRI,LTRI,IPUNCH,NEWBV
      COMMON/IWCRK/IBTRI(350)
      COMMON/IX/X(200)
      COMMON/IY/Y(200)
C
C     CALCULATE THE NUMBER OF POINTS IN EACH TRIANGLE AND ORDER THE
C     BOUNDARY POINTS.THE CODE IS AS FOLLOWS, THE VALUE OF IBTRI(K)
C     MILL TELL US HOW MANY BOUNDARY POINTS TRIANGLE ≠K≠HAS.
C
    5 FCRMAT(*1*,*WE HAVE FCUND A BAD TRIANGLE WITH IBTRI GT .3 IN
     HPROGRAM OUTSIDE, TRI=*,I4)
   10 FORMAT(* *,*THE GLOBAL LABELS ARE*,3I4)
   15 FORMAT(*1*)
  150 FORMAT(* *,*THIS LISTING IS DONE AFTER WE HAVE ELIMINATED TRIANGLE
     ≠S OUTSIDE OF CUR DOMAIN*)
   30 FORMAT(*0*,*TRIANGLE NO.*,10X,*VERTEX 1*,10X,*VERTEX 2*,10X,*VERTE
     *X 3*)
   90 FCRMAT(*0*)
   35 FCRMAT(* ● ,GX,I?,:3X,13,14X913,15X, I3)
  155 FCRMAT(*1*,*PROGRAM IS TERMINATED BECAUSE OF FAULTY BOUNDARY TRIAN
     ≠GLE*,/,* ● 9*I;TR: WAS GREATER THAN 3*)
      CALL FINDBP(IB,IP,IBTRI,NTRI,NBV)
C
C     HERE WE HAVE THE OPTICN TO ECHO CHECK THE BOUNDARY POINT SEQUENCES
C
      DO 1 I=1,NTRI
      IF(IBTRI(I).LT.4) GO TO 1
      WRITE(6,5)I
      WRITE(6,10)IP(1,I),IP(2,I),IP(3,I)
    1 CONTINUE
C
C
C      HERE WE MAKE ONE FINAL CHECK OF OUR BOUNDARY TRIANGLES.
C     IF IBTRI IS GREATER THAN 3, THE PROGAM IS KILLED.
C     ● *W@*IT Is SUGGESTED THAT IBVCHK=1 UNTIL ONE GETS PAST THIS POINT.
C     THIS HILL ALLOW ONE TO FIND THE BAD TRIANGLES.
C
      DO 68 I=1,NTRI
      IF(IBTRI(I).GT.3) GC TO 911
   68 CONTINUE
      Go TO 281
  911 WRITE(6,155)
      GO TO 754
  281 CONTINUE
C
C
C     THIS SECTION CHECKS TO SEE THAT ALL THE BOUNDARY TRIANGLES ARE
C     INSIDE THE DOMAIN
C

C     IF THERE ARE EXTRANEOUS TRIANGLES OUTSICE OF THE DOMAIN,THEY
C     HILL BE ELIMINATED ANO NTRI HILL BE ADJUSTED ACCORDINGLY
C
      CALL ELIM(IBTRI,IP,X,Y,NTRI)
C
C
  754 CONTINUE
      END
```

```
      OVERLAY(MODEL,7,0)
      PROGRAM TERM
      COMMON/NUMB/NVRTX,NTRI,LIST, IPUNCH,NEWBV
      COMMON/IHEIGHT/HEIGHT (200)
      COMMON/ICCNST/CONST1,CONST2
      COMMON/LSCOEF/SA(4) ,SD(4)
      COMMON/IALPHA/ALPHA(200)
      COMMON/IIP/IP(3,350)
      COMMON/IA/A(3,3)
      COMMON/IB/B(3)
      COMMON/IDEPTH/DEPTH(200)
      COMMON/WIND/TAUX,TAUY,CURL
      COMMON/IX/X(200)
      COMMON/IY/Y(200)
      COMMON/IALP/ALP(3)
      COMMON/IGRAD/DALPHAX,DALPHAY,DDEPTHX,DDEPTHY,DEX,DEY,AREA
      WRITE(6,1)
    1 FORMAT(*1*,*DYNAMIC BALANCE TERMS*)
      WRITE(6,2)
    2 FORMAT(*0*,*TRI*,8X,*BAROTROPIC TORQUE*, 8X,*BAROCLINIC TORQUE*,7X,
     ≠*CURL OF WIND*,5X,*BOTTOM FRICTION*)
      DO 999 I=1,NTRI
      J=IP(1,I)
      K=IP(2,I)
      L=IP(3,I)
      A(1,1)=X(J)
      A(1,2)=Y(J)
      A(1,3)=1.0
      A(2,1)=X(K)
      A(2,2)=Y(K)
      A(2,3)=1.0
      A(3,1)=X(L)
      A(3,2)=Y(L)
      A(3,3)=1.0
      CALL GRAD(DEPTH(J),DEPTH(K),DEPTH(L),DDEPTHX,DDEPTHY,CDEPTH)
      CALL ALPHX(I,DALPHAX,DALPHAY,ALPHA,SA,ALP)
      CALL GRAD(HEIGHT(J),HEIGHT(K),HEIGHT(L),DEX,DEY,CE)
      CALL DYBALAN(BRT,BRC,CURL,BFRIC,CONST1)
      WRITE(6,3)I,BRT,BRC,CURL,BFRIC
    3 FORMAT(* *,I3,11X,F10.4,14X,F10.4,12X,F10.4,9X,F10.4)
  999 CONTINUE
      END
```

```
      OVERLAY(MODEL,10,0)
      PROGRAM GRID
      COMMON/IN/N(300)
      COMMON/IGRAD/XMAX,YMAX,YMIN,RADIUS,XMIN,EXTRA(2)
      COMMON/IX/X (200)
      COMMON/IY/Y(200)
      COMMON/IHEIGHT/HEIGHT (z00)
      COMMON/IIP/IP(3,350)
      COMMON/NUMB/NVRTX,NTRI,LIST,IPUNCH,NEWBV
      COMMON/CALCCMP/XXMAX,YYMAX,ISTART,NBC
      COMMON/SCALES/USCALE,DSCALE,ALSCALE,G,E,Q,GAMMA,FO,EDDY
      DIMENSION ALAT(200),ALONG(200),LAT(200),LONG(200)
      EQUIVALENCE(HEIGHT(1),LONG(1))
      EQUIVALENCE (ALAT(1),Y(1))
      EQUIVALENCE(ALONG(1),X(1))
      EQUIVALENCE (IP(1,1),LAT (1))
      CALL CARTSN(LAT,ALAT,LONG,ALONG,NVRTX,ALSCALE,XXMAX,YYMAX,XMAX,YMA
     *X,YMIN,X,Y,RADIUS,XMIN)
      IPGRID=IPUNCH
C     THIS IS THE OPTION TO PUNCH THE COORDINATE OATA UP
C
      IF(IPGRID.EQ.0) GO TO 152
      WRITE(7,5)NVRTX
    5 FORMAT(15I5)
      DO 151 I=1,NVRTX
      WRITE(7,195)X(I),Y(I),N(I)
  195 FORMAT(F10.3,F10.3,50X,*STA.*,2X,I3)
  151 CONTINUE
      WRITE(7,140)YMIN,YMAX,YYMAX,XXMAX,RADIUS
  140 FORMAT(7F1O.2)
  152 CONTINUE
      END
```

```
      OVERLAY(MODEL,11,0)
      PROGRAM NRML
      COMMON/IN/N(300)
      COMMON/IALPHA/ALPHA(200)
      COMMON/IDELTA/DELTA(200)
      COMMON/IDEPTH/DEPTH(200)
      COMMON/WIND/TAUX,TAUY,CURL
      COMMON/NUMB/NVRTX,NTRI,LIST,IPUNCH,NEWBV
      CALL NORM(ALPHA,DELTA,DEPTH,TAUX,TAUY,CURL,NVRTX)
      IPNORM=IPUNCH
  250 FORMAT(F10.4,F15.4,F10.4,36X,*STA.*,1X,I3)
  260 FORMAT(*WIND STRESS AND CURL VALUES*,3F10.4)
      IF(IPNORM.EQ.0) GO TO 154
      00 153 I=1,NVRTX
      WRITE(7,250 )ALPHA(I) ,DELTA(I ),DEPTH(I),N(I)
  153 CONTINUE
      WRITE(7,260)TAUX,TAUY,CURL
  154 CONTINUE
      END
```

```
      OVERLAY(MCDEL,12,0)
      PROGRAM TRNS
      COMMON/ICCNST/C1,C2
      COMMON/IHEIGHT/HEIGHT (200)
      COMMON/NUMB/NVRTX,NTRI,LIST,IPTRANS,NEWBV
      COMMON/SCALES/USCALE,DSCALE,ALSCALE,G,E,Q,GAMMA,FO,EDDY
      CCMMON/IIP/IP(3,350)
      COMMON/IX/X(200)
      COMMON/IY/Y(200)
      COMMON/IALPHA/ALPHA(200)
      COMMON/IDELTA/DELTA(200)
      COMMON/IDEPTH/DEPTH(200)
      COMMON/IGRAD/DALPHAX,DALPHAY,DDEPTHX,DDEPTHY,DELEVX,DELEVY,AREA
      COMMON/WIND/TX,TY,CURL
      COMMON/LSCOEF/SA(4),SD(4)
      COMMON/IA/A(3,3)
      COMMON/IB/B(3)
      COMMON/IALP/ALP(3)
C
C     SUBROUTINE TO CALCULATE TRANSPORTS AT CENTROIC OF EACH TRIANGLE
C     CX AND CY ARE THE LOCATIONS OF TRIANGLE CENTERS
C     XTRANS AND YTRANSARE THE TRANSPORTS IN THE X AND Y DIRECTIONS
C     TTRANS IS THE TOTAL TRANSPORT
C
      WRITE(6,15)
   15 FORMAT(*1*)
      DIM=USCALE*DSCALE*Q
      WRITE(6,240)DIM
  240 FORMAT(* *,*TRANSPORT SCALE FACTOR IS*,F8.3,1X,*CUBIC METERS PER S
     *ECOND PER SQUARE METER ASSUMING DENSITY IS ONE.*)
      WRITE(6,230)
  230 FORMAT(**,*VELOCITY IS IN CENTIMETERS PER SECOND*,/)
      WRITE(6,180)
  180 FORMAT(* *,*TRIANGLE*,3X,*X-COOR*,4X,*Y-COOR* ,7X,*X-TRANSP*,7X,*Y-
     *TRANS*,5X,*TOT. TRANSP*,5X,*DEPTH*,6X,*U-ME AN*,4X,*V-MEAN*,5X,*V T
     *OT*,/)
C
      DO 100 I=1,NTRI
      J=IP(1,I)
      K=IP(2,I)
      L=IP(3,I)
C
      A(1,1)=X(J)
      A(1,2)=Y(J)
      A(1,3)=1.
      A(2,1)=X(K)
      A(2,2)=Y(K)
      A(2,3)=1.0
      A(3,1)=X(L)
      A(3,2)=Y(L)
      A(3,3)=1.
      CALL GRAD(HEIGHT(J),HEIGHT(K),HEIGHT(L),DELEVX,DELEVY,CELEV)
      CALL GRAD(DEPTH(J),DEPTH(K),DEPTH(L),DDEPTHX,DDEPTHY,CDEPTH)
```

```
      CALL ALPHX(I,DALPHAX,DALPHAY,ALPHA,SA,ALP)
C     DELTAGRADIENTS ARE HANDLED LIKE THE ALPHA GRADIENTS
      CALL ALPHX(I,DDELTAX,DDELTAY,DELTA,SD,ALP)
C
      CX=(X(J)+X(K)+X(L))/3.
      CY=(Y(J)+Y(K)+Y(L))/3.
C
      DEP=(DEPTH(J)+DEPTH(K)+DEPTH(L) )/3.
C
      XTRANS=DELEVY*DEP-C1*DDELTAY+C2*(DELEVY-DELEVX)-C1*C2*(DALPHAX-DAL
     ≠PHAY)+TY
C
      YTRANS=-(DELEVX*DEP)+C1*DDELTAX-C2*(DELEVY+DELEVX)-C1*C2*(DALPHAY+
     ≠DALPHAX)-TX
C
      TRANS=(XTRANS*XTRANS+YTRANS*YTRANS)**.5
C
      DEP=-DEP*200.
      XTRANS=XTRANS*DIM
      YTRANS=YTRANS*DIM
      TRANS=TRANS*DIM
      U=XTRANS/DEP*100.
      V=YTRANS/DEP*100.
      VT=TRANS/DEP*100.
      WRITE(6,185)I,CX,CY,XTRANS,YTRANS,TRANS,DEP,U,V,VT
  185 FORMAT(3X,I3,5X,F6.2,4X,F6.2,4X,F11.4,4X,F11.4,4X,F11.4,3X,F7.1,3X
     ≠,F9.4,2X,F9.4,2X,F9.4)
      IF(IPTRANS.EQ.0)GO TO 100
      WRITE(7,260)XTRANS,YTRANS,TRANS,U,V,VT,I
  260 FORMAT(6F10.3,10X,*TRANS*,2X,I3)
  100 CONTINUE
  101 CONTINUE
      END
```

```
      OVERLAY(MODEL,13,0)
      PROGRAM VELO
      COMMON/IHEIGHT/HEIGHT(200)
      COMMON/NUMB/NVRTX,NTRI,LIST,IPVELO,NEWBV
      DIMENSION ALPH(3)
      COMMON/IGRAD/DALPHAX,DALPHAY,DDEPTHX,DDEPTHY,DEX,DEY,AREA
      COMMON/LSCOEF/SA(4),SD(4)
      COMMON/SCALES/USCALE,DSCALE,ALSCALE,G,E,Q,GAMMA,FO,EDDY
      COMMON/IIP/IP(3,350)
      COMMON/IA/A(3,3)
      COMMON/IB/B(3)
      COMMON/IALPHA/ALPHA(200)
      COMMON/IDEPTH/DEPTH(200)
      COMMON/WIND/TAUX,TAUY,CURL
      COMMON/IX/X(200)
      COMMON/IY/Y(200)
      WRITE(6,1)
    1 FORMAT(*1*,*COMPARATIVE VELOCITIES IN CM/SEC*)
      WRITE(6,2)
    2 FORMAT(*0*,16X,*EKMAN*,26X,*BAROTROPIC*,25X,*SURFACE*,26X,*BOTTOM*
     ≠,/)
      WRITE(6,4)
    4 FORMAT(* *,*TRI*,5X,*U*,9X,*V*,7X,*TOTAL*,10X,*U*,9X,*V*,7X,*TOTAL
     ≠*,8X,*  u  *,5X,*  v  *,5X,*TOTAL*,10X,*U*,9X,*V*,7X,*TOTAL*,/)
C
      DO 999 K=1,NTRI
      I=IP(1,K)
      J=IP(2,K)
      L=IP(3,K)
      A(1,1)=X(I)
      A(1,2)=Y(I)
      A(1,3)=1.
      A(2,1)=X(J)
      A(2,2)=Y(J)
      A(2,3)=1.
      A(3,1)=X(L)
      A(3,2)=Y(L)
      A(3,3)=1.
C
      CALL GRAD(DEPTH(I),DEPTH(J),DEPTH(L),DDEPTHX,DDEPTHY,CDEPTH)
      CALL ALPHX(K,DALPHAX,DALPHAY,ALPHA,SA,ALPH)
      CALL GRAD(HEIGHT(I),HEIGHT(J),HEIGHT(L),DEX,DEY,CE)
      CALL EKMAN(UE,VE,TOTE)
      CALL BAROT(UB,VB,TOTB)
      CALL SURF(UE,VE,UB,VB,US,VS,TOTS)
      CALL BOTT(UBOT,VBOT,TOTBOT)
      WRITE(6,5)K,UE,VE,TOTE,UB,VB,TOTB,US,VS,TOTS,UBOT,VBOT,TOTBOT
    5 FORMAT(* ● ,I3,F8.3,2X,F8.3,2X,F8.3,5X,F8.3,2X,F8.3,2X,F8.3,5X,F8.3
     ≠,2X,F8.3,2X,F8.3,5X,F8.3,2X,F8.3,2X,F8.3)
      IF(IPVELO.EQ.0) GO TO 952
      WRITE(7,105)K,US,VS,TOTS,UBOT,VBOT,TOTBOT
  105 FORMAT(I5,6F10.3,2X,*VELOCITY*)
  952 CONTINUE
  999 CONTINUE
      END
```

```
      OVERLAY(MODEL,14,0)
      PROGRAM BCHX
C
C          PROGRAM TO APPROXIMATEBOUNDARY CONDITIONS BY SOLVING
C     BOTTOM FRICTIONLESS CASE.
C
      COMMON/EXTRA/XX(100),YY(100) ,00(100)
      COMMON/IGRAO/IFLG,IBEGIN,IEND,NIBP,NM
      COMMON/IIP/IP(3,350)
      COMMON/IX/X(200)
      COMMON/IY/Y(200)
      COMMON/ICCNST/CONST1,CONST2
      COMMON/IDEPTH/DEPTH(200)
      COMMON/IALPHA/ALPHA(200)
      COMMON/BOUND/IB(75)● BV{751?NBV
      COMMON/WIND/TAUX,TAUY,CURL
      COMMON/NUMB/NVRTX,NTRI,NOINTG,IPINTG,NEWBV
      COMMON/IA/A(3,3)
      COMMON/IB/B(3)
      COMMON/IN/N(300)
      COMMON/CUTOFF/IBP,NFLX,J1,J2,DEEP
      COMMON/SCALES/USCALE,DSCALE,ALSCALE,G,E,Q,GAMMA,FO,EDDY
      COMMON/LSCOEF/SA(4),SD(4)
      CCMMON/IHEIGHT/XI(100),YI(100)
      COMMON/IRHS/DST(100) ,EI(1OO)
C
      ISLAND=IBP
      NIBP=IBP
      LANO=NFLX
      ESCALE=FO*USCALE*ALSCALE*Q*100./G
      IF(NEWBV.GT.1) GO TO 198
      IF(NOINTG.EQ.0) GO TO 201
C
C     ENTER OVERLAY TO OBTAIN ELEVATION CHANGES ALONG OEPTH CONTOURS*
C
      CALL OVERLAY(5HMODEL,12,1)
      GO TC 207
  198 00 199 I=1,NBV
      XI(I)=XX(I)
      YI(I)=YY(I)
  199 EI(I)=BV(I)+DD(I)
      GO TO 207
  201 DO 206 I=1,NBV
      J=IB(I)
      READ(5, 200)X,N(J),XI(I) ,YI(I),DH
      XX(I)=XI(I)
      YY(I)=YI(I)
      DH=DH/ESCALE
      DD(I)=DH
  206 EI(I)=BV(I)+DH
  207 CONTINUE
      IBP=ISLAND
      NFLX=LANO
```

```
      NIBP=ISLAND
C
C            RESULTS FROMSCLVER ARE LISTED HERE.
      IF(NEWBV.GT.1) GO TO 203
      CALL WHIN(DST)
C
      WRITE(6,25)
      DO 21 I=1,NBV
      BVV=BV(I)*ESCALE
      J=IB(I)
      DH=0.
      DP=DEPTH(J) *DSCALE
      CALL BYPASS(I,IYES)
      IF(IYES.EQ.1) GO TO 19
      CALL WHOUT(DST, XI(I),YI(I),EDST)
      EEI=EI(I)*ESCALE
      DH=EEI-BVV
      GO TO 121
   19 EDST=0.0
      EEI=BVV
  121 IF(IPINTG.EQ.0) GO TO 202
      WRITE(7,200) I,N(J),XI(I) ,YI(I),DH
  202 CONTINUE
   21 WRITE(6,30) I,J,N(J),BVV,DP,X (J),Y(J) ,DST(I),XI(I),YI(I),EDST,EEI,D
     ≠H
  203 CONTINUE
C
C            THISIS WHERETHE BOUNDARY ELEVATIONS ARE ALTERED ACCORDING
C     TO THE RESULTS FROMSCLVER.
C
      CALL ALTER(XI,YI)
C
C     THE NEW BOUNDARY  ARE  LISTED HERE.
C
      WRITE(6,35)
      DO 41I=1,NBV
      J=IB(I)
      BVV=BV(I)*ESCALE
   41 WRITE(6,40)I,J,N(J) ,BVV
C
C     HERE THE NEW BOUNDARY VALUES ARE PLOTTED UP AS A FUNCTION OF
C     DISTANCE ALONG THE BOUNDARY.
C
C
      CALL OVERLAY(5HMODEL,12,2)
C
    5 FCRMA?I3F1O,2I
   10 FORMAT(3I5,55X,*TRIA.*, 15)
   15 FORMAT(I5,F10.2)
   25 FORMAT(1H1,*BND.*,3X,*GLB.*,3X,*STA.*,4X,*ELEV.*,6X,*DEPTH*,5X,*EN
     ≠TR. COOR.(DIST.)*,5X,*EXIT COOR.(DIST.)*,5X,*EXIT ELEV.*,5X,*ELEV.
     ≠CHNG.*,/)
   30 FORMAT(1H ,I3,3X,I4,3X, I3,5X,F6.2 ,4X,F6. 0,5X,F5.2,*,* ,F5.2,*(*,F5.
     ≠2,*)*,5X,F5.2,*,*,F5.2,*(*,F5.2,*)*,5X,F8.4,7X,F8.4)
   35  FORMAT(*1*,*BND. NO.*,5X,*GLB. NO.*,5X,*STA. NO.*,5X,*NEW ELEV.*,/
     ≠)
   40  FORMAT(* *,I5,8X,I5,8X,I5,8X,F9.4)
  200 FORMAT(2I5, 2F10.3,2F10.4)
  205 FORMAT(F10.4,15X,F10.4)
  210 FORMAT(47X,F10.4)
  215 FORMAT(F15.7)
  220 FORMAT(11X,4F10.4)
  305 FORHAT( 1X,FIO.2SF1OO2,54X,I5 )
  310 FORMAT(2F10.2)
      END
```

```
      OVERLAY(MODEL,14,1)
      PROGRAM SLVBV
      COMMON/IGRAC/IFLG,IBEGIN,IEND,NIBP,NM
      COMMON/IIP/IP(3,350)
      COMMON/IX/X(200)
      COMMON/IY/Y(200)
      COMMON/ICONST/CONST1,CONST2
      COMMCN/IDEPTH/DEPTH(200)
      COMMON/IALPHA/ALPHA(200)
      COMMON/BOUND/IB(75),BV(75),NBV
      COMMON/WIND/TAUX,TAUY,CURL
      CCMMCN/NUMB/NVRTX,NTRI,NOINTG,IPINTG,NEWBV
      COMMCN/IA/A(3,3)
      COMMON/IB/B(3)
      COMMON/CUTOFF/IBP,NFLX,J1,J2,DEEP
      COMMON/SCALES/USCALE,DSCALE,ALSCALE,G,E,Q,GAMMA,FO,EDDY
      COMMON/LSCOEF/SA(4),SD(4)
      COMMON/IHEIGHT/XI(100),YI [100)
      COMMON/IRHS/DST(100),EI(100)
C
C
C          HERE WE BEGIN BY IDENTIFYING EACH BOUNDARY POINT ANO ITS
C     VALUE.  THEN WE ENTER THIS INFORMATION INTO SOLVER WHICH SOLVES
C     THE FIRST ORCEREQUATION ANO RETURNS THE ELEVATION AT THE EXIT
C     PCINTFCR THE DEPTH INTEGRATED ALONG.
      ISLAND=IBP
c
      DO 7 I=1,NBV
      J=IB(I)
C
c     INITIALIZE SOME PARAMETERS
C
      XI(I)=0.
      YI(I)=0.
C
C          NOTE THATIF WE ARE BELOW THE CUTOFF DEPTH, SURFACE
C     ELEVATIONSALONG THE BOUNDAYWILL BE CALCULATED BY DELTA-DS.
C     ALSOIi WE ARE AT THE SHALLOH WATER CUTOFF DEPTH, SOLVER WILL NOT
C     BE CALLED.
C
      CALL BYPASS(I,IYES)
      IF(IYES.EQ.1) GO TO 7
      XX=X(J)
      YY=Y(J)
      BVV=BV(I)
      CALL SOLVER(BVV,XX,YY,ELEV,X1,Y1,J,SA)
      XI(I)=X1
      YI(I)=Y1
      EI(I)=ELEV
      IBP=ISLAND
    7 CONTINUE
      END


      OVERLAY(MODEL,14,2)
      PROGRAM TPLOT
      COMMON/IRHS/DST(100),EI(100)
      COMMON/SCALES/USCALE,DSCALE,ALSCALE,G,E,Q,GAMMA,FO,EDDY
      COMMON/BOUND/IB(75),BV(75),NBV
      ESCALE=FO*USCALE*ALSCALE*Q*100./G
      WRITE(6,45)
   45 FORMAT(1H1)
      0 0 1 I=1,NBV
      BV(I)=BV(I)*ESCALE
    i DST(I)=DST(I)*10.
C
C
c     CALL PRINTER PLOT RCUTINE
      CALL FPLOT(EV,DST,NBV,1.0,10.0,20.0,09,6,2,-1.0)

      00 2 I=1,NBV
    2 BV(I)=BV(I)/ESCALE
      END
```

```
      SUBROUTINE NEWBVAL
C
C     SUBROUTINE TO READ IN NEW BOUNDARY VALUES.
C
      COMMON/IRHS/RHS(200)
      COMMON/BOUND/IB(75),BV(75),NBV
      COMMON/SCALES/USCALE,DSCALE,ALSCALE,G,E,Q,GAMMA,FO, EDDY
      COMMON/WIND/TAUX,TAUY,CURL
      COMMON/CUTOFF/NIBP,NFLX,NOMAT,JJ
      ESCALE=G/(FO*USCALE*ALSCALE*Q*100.)
C
    5 FORRAT(I5,F1O*4)
   10 FORMAT(3F10.4)
C
      READ (5,1O)TAUX,TAUY,CURL
      WSCL=FO*USCALE*DSCALE*Q*10000.
      TAUX=TAUX/WSCL
      TAUY=TAUY/WSCL
      CURL=CURL/(WSCL*ALSCALE*100.)
C
C     WE HAVE JUST READ IN NEW WIND STRESS VALUESO
C
      DO 1 I=1,NBV
      READ(5,5)IBN,BV(I)
      BV(I)=BV(I)*ESCALE
    1 CONTINUE
C
C     WE HAVE READ IN THE NEW BOUNDARY VALUES ALONG WITH THEIR BOUNDARY
C     NUMBERS, THEN PROCEEDED TO NONDIMENSIONALIZED THEM.
C
C     WE NOW RETURN TO REASSEMBLE THE RIGHT HANO SIDE AND USE THE NEW
C     BOUNDARY VALUES.  THIS IS DONE BY SETTING NOMAT=-1.
C
      JJ=1
C
      RETURN
      END
```

```
      SUBROUTINE TRIAREA(A,AREA)
      DIMENSION A(3,3)
      AA=A(1,1)*(A(2,2)*A(3,3)-A(2,3)*A(3,2))
      BB=-A(1,2)*(A(2,1)*A(3,3)-A(2,3)*A(3,1))
      CC=A(1,3)*(A(2,1)*A(3,2)-A(2,2)*A(3,1))
      AREA=.5*(AA+BB+CC)
      RETURN
      END
```

```
      SUBROUTINE SOLVE(A,B)
      DIMENSION A(3,3),B(3),C( 3),X(3)
      CALL TRIAREA(A,AREA)
      DO 2 J=1,3
      DO 1 I=1,3
      K=J-1
      IF(J.NE.1) A(I,K)=C(I)
      C(I)=A(I,J)
      A(I,J)=B(I)
    1 CONTINUE
      CALL TRIAREA(A,X(J))
    2 CONTINUE
      A(1,3)=C(1)
      A(2,3)=C(2)
      A(3,3)=C(3)
      B(1)=X(1)/AREA
      B(2)=X(2)/AREA
      B(3)=X(3)/AREA
    8 RETURN
      END
```

```
      SUBROUTINE GRAD (E,F,G,DX,DY,C)

      SUBROUTINE TO CALCULATE GRADIENTS
      E,F, AND G ARE THE VALUES TO BE FILLED INTO THEB VECTOR
      DX,DY AND C ARE THE SOLUTIONS TO BE RETURNEO

      COMMON/IA/A(3,3)
      COMMON/IB/B(3)
      B(1)=E
      B(2)=F
      B(3)=G
      CALL SOLVE(A,B)
      DX=B(1)
      DY=B(2)
      C=B(3)
      RETURN
      ENO
```

```
      SUBROUTINE BETA1(YMIN,YMAX,FO)
C
C
C     THIS SUBROUTINE CALCULATES THE MEANCCRIOLISVALUE FoR THE REGION
C     FO IS THE HEAN CORICLIS VALUE
C
      OMEGA=.000072722052
C
      DEG=YMIN
      FO=2.*(OMEGA)*SIN(DEG)
      DEG=YMAX
      FM=2.*(OMEGA)*SIN(DEG)
      FC=(FC+FM)/2.
      RETURN
      END
```

```
      SUBROUTINE ALPHX(I,DALPHAX,DALPHAY,ALPHA,S,ALPH)
      DIMENSION ALPH(3),ALPHA(1),S(1)
      COMMON/IIP/IP(3,350)
      COMMON/IDEPTH/DEPTH(200)
      COMMON/IX/X(200)
      COMMON/IY/Y(200)
      COMMON/IA/A(3,3)
      COMMON/IB/B(3)
C
C     "SUBROUTINE TO GET ALPHA AND DELTA GRADIENTS BY USING A THIRD ORDER
C          LEAST SQUARES FIT TO THE ALPHA AND DELTA FIELDS.
C
      J=IP(1,I)
      K=IP(2,I)
      L=IP(3,I)
C
      RDEP=(DEPTH(J)+DEPTH(K)+DEPTH(L))/3.
C
C     CHECK TO SEE IF ALL THE DEPTHS ARE EQUAL
C
      IF(DEPTH(J).EQ.DEPTH(K).AND.DEPTH(J).EQ.DEPTH(L)) GO TO 3
      GO TO 4
    3 RDEP=DEPTH(J)
      ALPH(1)=ALPHA(J)
      ALPH(2)=ALPHA(K)
      ALPH(3)=ALPHA(L)
      GO TO 5
    4 CONTINUE
C
C     IFDEPTHS ARE NOT EQUAL, USE LSF FUNCTION TO GET ALPHA OR DELTA
C          AT REFERENCE DEPTH.
C
      DO 1 M=1,3
      N=IP(M,I)
      D0=DEPTH(N)
      DZ=RDEP-D0
      D1=3.*S(1)*D0*D0+2.*S(2)*D0+S(3)
      D2=6.*S(1)*D0+2.*S(2)
      D3=6.*S(1)
      ALPH(M)=ALPHA(N)+D1*DZ+D2*(DZ*DZ/2.)+D3*(DZ*DZ*DZ/6.)
    1 CONTINUE
C
C     RESET A, THE POSITIONMATRIXAND GET HORIZONTAL GRADIENTS
C
    5 CONTINUE
      A(1,1)=X(J)
      A(1,2)=Y(J)
      A(1,3)=1.
      A(2,1)=X(K)
      A(2,2)=Y(K)
      A(2,3)=1.
      A(3,1)=X(L)
      A(3,2)=Y(L)
      A(3,3)=1.
C
      CALL GRAD(ALPH(1),ALPH(2),ALPH(3),DALPHAX,DALPHAY,CALPHA)
      RETURN
      END
```

```
      SUBROUTINE INVR ( A, N, B, M, DETERM, ISIZE, JSIZE )
C
C     NAME:        INVR
C     PURPOSE:     LINEAR EQUATION SOLUTION (MATRIX INVERSION)
C     ALGORITHM:   GAUSS-JORDAN ELIMINATION, FULL PIVOT SEARCH.
C                  SEE ANY NUMERICAL ANALYSIS TEXT FOR REFERENCE.
C     AUTHOR:      ORIGINAL VERSION, CIRCA 1966, AUTHOR UNKNOWN.
C                  RECOPIED, 1976, RIK LITTLEFIELD (U.W.), WITH
C                  STYLISTIC CHANGES AND MINOR BUG CORRECTIONS (NO
C                  ALGORITHM OR CALLING SEQUENCE CHANGES)
C     USAGE:       SEE U.WASH. COMPUTING INFORMATION CENTER DOCUMENT
C                  NUMBER W00042 FOR FULL DESCRIPTION.  BASICALLY,
C                    A = INPUT: COEFFICIENT MATRIX, LOGICALLY (N,N),
C                        PHYSICALLY (ISIZE,*).  OUTPUT: A-INVERSE.
C                    B = INPUT: RIGHT-HAND SIDE, LOGICALLY (N,M),
C                        PHYSICALLY (ISIZE,*).  OUTPUT: SOLUTION MATRIX.
C                    DETERM = DETERMINANT OF A, 0. IF A APPEARS SINGULAR.
C                    JSIZE = CURRENTLY UNUSED, ORIGINALLY 2ND DIM. OF A.
C     LIMITS:      (100,100) SYSTEM.  CHANGE DIMENSION STATEMENTS FOR
C                  LARGER SYSTEMS.
C     TIMING:      ORDER(N**3).  TYPICAL .4 TO .5 SECONDS FOR (20,20),
C                  M=0, USING *RUN* FORTRAN COMPILERS CDC 6400.
C     COMMENTS:    THIS PARTICULAR IMPLEMENTATION OF THE ALGORITHM IS NOT
C                  THE BEST POSSIBLE - SEE ANY MATH SUBROUTINE LIBRARY
C                  (E.G.,IMSL) FOR IMPROVED ROUTINES.
C
      REAL      A(ISIZE,1), B(ISIZE,1)
      INTEGER PIVOT(100), INDEX(100,2)
      INTEGER COLUMN,ROW
      DO 10 J = 1,N
      DETERM=1.
   10    PIVOT(J) = 0
      DO  130 I = 1,N
C
C       *** FULL SEARCH FOR PIVOT ELEMENT (BRANCH OUT IF NO NON-ZERO
C       *** PIVOT IS FOUND)
C
         AMAX = 0.0
         DO 30 I1 = 1,N
            IF ( PIVOT(I1).NE.0 ) GO TO 30
            DO 20 J = 1,N
               IF (PIVOT(J).NE.0 ) GO TO 20
                  IF (ABS(A(I1,J)) .LE. AMAX ) GO TO 20
                     ROW = I1
                     COLUMN = J
                     AMAX = ABS(A(I1,J))
   20          CONTINUE
   30       CONTINUE
         IF ( AMAX .EQ.0.0 ) GO TO 200
         PIVOT(COLUMN) = 1
C
C       *** INTERCHANGE ROWS TO PUT PIVOT ELEMENT ON DIAGONAL
C
```

```
         IF  (  ROH .EQ.COLUMN ) GO TO GO
            DETERM = -DETERM
            DO  40 J = 1,N
               SWAP = A(RCH,J)
               A(ROH,J) = A(COLUMN,J)
   40          A(COLUMN,J) = SWAP
            IF  (M.LE.0) GO TO  60
            DO  50  J  =  1,M
               SWAP =  B(ROH,J)
               B(ROH,J)  = B(COLUMN,J)
   50          B(COLUMN,J)  =  SWAP
   60    INDEX(I,1)  =  RCH
         INDEX(I,2)  =  COLUMN
         PVTELM = A(COLUMN,COLUMN)
         DETERM  =  DETERM*PVTELM
C
C       *** NORMALIZE PIVCT ROW (DIVIDE BY PIVOT ELEMENT)
C
         A(COLUMN,COLUMN)  =  1.0
         00  70  J = 1,N
   70       A(COLUMN,J)  =  A(COLUMN,J)/PVTELM
         IF ( M.LE.0  ) GC TC  90
         DC  80  J  =  1,M
   80       B(CCLUMN,J)  =  B(COLUMN,J)/PVTELM
C
C       ***  REDUCE  NON-PIVOT ROWS
C
   90    DC 120 11  =  1,N
            IF   (I1.EQ.COLUMN) GO TO 120
               T  =  A(I1,COLUMN)
               A(I1,COLUMN)  =  0.0
               00  100  J  =  1,N
  100           A(I1,J)   =   A(I1,J)   A(COLUMN,J)*T
               IF (M.LE.0)  GO   TO   120
               00   110 J = 1,M
  110             B(I1,J) = B(I1,J)    B  (COLUMN,J)*T
  120       CONTINUE
  130    CONTINUE
C
C       *** Elimination DONE, INTERCHANGE COLUMNS TO COMPLETE INVERSE
C
      00 150 J  =  1,N
         L  =  N+1-J
         IF   (   INDEX(L,1).EQ.INDEX(L,2) ) GO  TO  150
            ROW = INDEX(L,1)
            COLUMN  =  INDEX(L,2)
            00  140  I  =  1,N
               SWAP  =  A(I,RCW)
               A(I,ROW)  =  A(I,COLUMN)
  140          A(I,COLUMN)  =  SWAP
  150    CONTINUE
      RETURN
C
C    *** SINGULAR MATRIX - ERROR EXIT
C
  200 DETERM  =  000
      RETURN
      ENO
```

```
      SUBROUTINE SMOOTH(VAL,S,D,SD,NPTS)
      COMMON/IDEPTH/DEP(200)
      DIMENSION VAL(1),S(1)
C
C     SUBROUTINE TO SMOOTH THE DATA ACCORDING TO LEAST SQUARES FIT,
C
      DO 1 I=1,NPTS
      V=S(1)*DEP(I)*DEP(I)*DEP(I)+S(2)*DEP(I)*DEP(I)+S(3)*DEP(I)+S(4)
      DD=VAL(I)-V
      IF(DD.LT.-D) DD=-D*SD
      IF(DD.GT.D) DD=D*SD
      VAL(I)=DD+V
    1 CONTINUE
      RETURN
      END

      SUBROUTINE SETMAT(NVRTX,IFILE,MASL)
      COMMON/IWORK/VALP(4500)
      COMMON/INT/INTP(4500)
      COMMON/IRHS/RHS(200)
C
C     THIS SUBROUTINE SETS THE MATRIX AND RIGHT HAND SIOE TO O        OPERATC
C
      IF(IFILE.EQ.1) GO TO 15
      00 99 I=1,4500
      VALP(I)=0.
   99 INTP(I)=0
   15 00 16 I=1,200
   16 RHS(I)=0.
   17 CONTINUE
      NROW=NVRTX
      NCOL=NVRTX
      MSZ=4500
C
C     HERE HE TELL THE SOLVING ROUTINE THAT WE ARE READY
C
      CALL MINIT(INTP,NROW,NCOL,MSZ,MASL)
      RETURN
      ENO

      SUBROUTINE NORM(ALPHA,DELTA,DEPTH,TAUX,TAUY,CURL,NVRTX)
      COMMON/SCALES/USCALE,DSCALE,ALSCALE,G,E,Q,GAMMA,FO,EDDY
      DIMENSION ALPHA(1),DELTA(1),DEPTH(1)
C
C     SUBROUTINE USEO FOR NORMALIZATION OF STATION CATA
C     ALPHA IS DENSITY INTEGRATE VERTICALLY (GM/CM**3)*M
C     DELTA IS ALPHA INTEGRATED VERTICALLY (GM/CM**3)*M**2
C     DEPTH IS DEPTH(M)
C     TAUX AND TAUY ARE HINO STRESS VALUES IN THE X AND Y DIRECTION
C        WIND STRESS UNITS ARE DYNES/CM*CM
C     DSCALE IS THE DEPTH SCALE (M)
C     USCALE IS THE HORIZONTAL VELOCITY SCALE (M/SEC)
C     FO IS THE AVERAGE CORIOLIS PARAMETER
C     Q ANO E ARE THE MEAN ANO PERTURBATION DENSITY RESPECTIVELY
C     NVRTX IS THE NUMBER OF STATIONS
C
C     TO CALCULATE SCALE FACTORS
      ALPH=E*DSCALE
      DELT=ALPH*DSCALE
      DO 12 I=1,NVRTX
      ALPHA(I)=(ALPHA(I)+DEPTH(I)*Q)/ALPH
      DELTA(I)=(DELTA(I)-(DEPTH(I)*DEPTH(I)*Q)/2.)/DELT
      DEPTH(I)=DEPTH(I)/DSCALE
   12 CONTINUE
C
      STRESS=FO*USCALE*DSCALE*10000.*Q
      TAUX=TAUX/STRESS
      TAUY=TAUY/STRESS
      STRESS=STRESS/(100.*ALSCALE)
      CURL=CURL/STRESS
      RETURN
      E NO
```

```
      SUBROUTINE CARTSN(LAT,ALAT,LONG,ALONG,NVRTX,ALSCALE,XXMAX,YYMAX,XM
     *AX,YMAX,YMIN,X,Y,RADIUS,XMIN)
      DIMENSION LAT(1),ALAT(1),LONG(1),ALONG(1),X(1),Y(1)
      CALL RADIANS(NVRTX,LAT,ALAT,LONG,ALONG,YMIN,XMIN,YMAX,AVEL,X,Y)
      CALL MERCTR(X,Y,ALSCALE,NVRTX,XMIN,YMIN,AVEL,RADIUS)
      YYMAX=0.
      XXMAX=0.
      DO 18 I=1,NVRTX
      YYMAX=AMAX1(YYMAX,Y  (I))
      XXMAX=AMAX1(XXMAX,X(I))
   18 CONTINUE
      RETURN
      END



      SUBROUTINE RADIANS(NVRTX,LAT,ALAT,LONG,ALONG,YMIN,XMIN,YMAX,AVEL,X
     *,Y)
C
C     THIS SUBROUTINE CONVERTS DEGREES LATITUDE ANDLONGTUDE INTO
C     RADIANS.  IT ALSO RETURNS THEMINIMUMLATIDUDE TO BE USED ASTHE
C     Y=0REFERENCE AND THE MAXIMUM LONGITUDE(H) TO BE USED AS THE X=0
C     REFERENCE.  THE AVERAGE LATITUDE IS ALSO RETURNED.
C
      DIMENSION LAT(1),LONG(1),ALAT(1),ALONG(1),X(1),Y(1)
      RADIAN=3.141492654/180.
      AVEL=0.
      DO 1 I=1,NVRTX
      DEG=ALAT(I)/60.
      ALAT(I)=FLOAT(LAT(I))+DEG
      ALAT(I)=ALAT(I)*RADIAN
      AVEL=AVEL+ALAT(I)
      DEG=ALONG(I)/60.
      ALONG(I)=FLOAT (LONG(I)) +DEG
    1 ALONG(I)=ALONG (I)*RADIAN
      AVEL=AVEL/FLOAT(NVRTX)
      YMIN=ALAT(1)
      XMIN=ALONG(1)
      YMAX=ALAT(1)
      DO 2 I=2,NVRTX
      YMIN=AMIN1 (YMIN,ALAT(I) )
      XMIN=AMAX1(XMIN, ALONG(I ))
    2 YMAX=AMAX1(YMAX,ALAT(I))
      RETURN
      END



      SUBROUTINE MERCTR(X,Y,ALSCALE,NVRTX,XMIN,YMIN,AVEL, RADIUS)
C
C     THIS Subroutine DOES THE MERCATOR TRANSFORMATION.   THE DISTANCES
C     ARESCALEDBY THE LENGTH SCALE(ALSCALE) AND THE AVERAGE DISTORTION
C     FACTOR SUCH THAT THE UNIT LENGTH IN X ANO Y IS APPROXIMATELY ONE
C     HORIZONTAL LENGTH SCALE.
C
      DIMENSION X(1),Y(1)
      RADIUS=6378000./ALSCALE*COS(AVEL)
      XSHIFT=XMIN*RADIUS
      ARG=YMIN/2.+.7853982
      ARG=TAN(ARG)
      YSHIFT=RADIUS*ALOG(ARG)
      DO 1 I=1,NVRTX
      X(I)=-(RADIUS*X(I)-XSHIFT)
      Y(I)=RADIUS*ALOG(TAN(Y(I)/2.+.7853982))-YSHIFT
    1 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE  HESH
C
C     HERE WE GENERATE THE MESH.
C
      COMMON/IWCRK/P(204,2),VERT (402,6)
      COMMON/INT/ISIOE(607,2),ITRI (403,3)
      COMMON/IX/X(200)
      COMMON/IY/Y(200)
      COMMON/IIP/IP(3,350)
      COMMON/NUMB/NVRTX,NTRI,LIST,NIBP,NEWBV
      COMMON/BOUND/IB(75),BV(75),NBV
      COMMON/CALCOMP/XXMAX,YYMAX,ISTART,NBC
C
      0 0 101 I=1,NVRTX
      P(I,1)=X(I)
      P(I,2)=Y (I)
  101 CONTINUE
      CALL TRIAN
      DO 2 I=1,NTRI
      IS1=ITRI(I,1)
      IS2=ITRI(I,2)
      IS3=ITRI(I,3)
      JP1=ISIDE(IS1,1)
      JP2=ISIOE(IS1,2)
      JP3=ISIDE(IS2,1)
      IF(JP1. EQ.JP3.OR.JP2.EQ.JP3) JP3=ISIDE(IS2,2)
      IP(1,I)=JP1
      IP(2,I)=JP2
      IP(3.I)=JP3
    2 CONTINUE
C
      RETURN
      END


      SUBROUTINE TRIAN
C
C     THE INFCRMATICN NEED TO RUN THE PROGRAM IS ASFOLLOWS,
C     AN INTEGER ARRAY T WHICH KEEPS TRACK OF WHAT THREE SIDES EACH
C          TRIANGLE HAS.  THIS  MUST BE A NTRI BY 3 ARRAY WHERE NTRI IS
C          THE MAXIMUM NUMBER OF TRIANGLES EXPECTED.
C     AN INTEGER  ARRAY,S, WHICH RECORDS THE ENDPOINTS OF EACH LINE.
C          THIS MUST BE AN NSIDE BY 2 ARRAY WHERE NSIDE IS THE MAXIMUM
C          NUMBER CF SIDES ONE EXPECTS.
C     NP, THE NUMBER OF POINTS ONE HAS,
C     NIBP, THE NUMBER OF INTERIOR BOUNDARY POINTS,
C     A REAL ARRAY,P, WHICH  CONTAINS  THE X ANO Y COORDINATES OF EACH
C          POINT.  THE X COORDIANTE GOES INTO THE P(NP,1) POSITION
C          ANO  THE Y COORDIANTE GOES INTO  THE P(NP,2) POSITION.
C          THEREFORE8 P IS AN NP BY 2 ARRAY WHERE NP IS THE NUMBER OF
C          POINTS.
C     ALSC NEED  ARE TWO WORKING ARRAYS, VERT(NT,3) ANO IS(3)
C          VERT HILL STORE THE COORIDIANTES OF EACH VERTEX OF EACH
C          TRIANGLES AS THEY ARE GENERATED.  IS WILL CONTAIN THE SIDES.
C     *************************NOTE*********************
C     IF INTERIOR BOUNDARY POINTS ARE ENTERED (ISLAND), THEN THEY MUST
C          BE THE FIRST  DATA CARDS REAO IN AND THEY HUST BE REAO IN
C          CLOCK WISE ORDER.
C     THE EXTERIOR BOUNDARY STATION NUMBERS ARE REAO IN AFTER THE
C          X ANO Y COORDINATES OF EACH STATION HAS BEEN ENTERED, ANO
C          THE EXTERIOR BOUNDARY STATION S ARE REAO IN COUNTERCLOCK
C          WISE ORDER.
C
      INTEGER S,T
      DIMENSION IS(3)
      COMMON/IWORK/P(204,2),VERT(402,6)
      COMMON/INT/S(607,2),T(403,3)
      COMMON/NUMB/NP,NT,LIST,NIBP,NEWBV
      COMMON/CALCOMP/XMAX,YMAX,ISTART,NBC
      COMMON/BOUND/IB(75),BV(75),NBV
C
```

```
C     THE FIRST STEP IS TO FIND THE MAXIMUM DOMAIN COVERE O BY THE POINTS
      XMIN=0.
      YMIN=0.
C
      IF(XMIN. NE. XMAX) GO TO 101
      XMAX=P(1, 1)
      XMIN=XMAX
      YMAX=P(1,2)
      YMIN=YMAX
      DO 1 I=2,NP
      IF(P(I,1).GT.XMAX)XMAX=P(I,1 )
      IF(P(I, 1).LT.XMIN)XMIN=P (I,1)
      IF(P(I,2).GT.YMAX)YMAX=P(I,2)
      IF(P(I,2).LT.YMIN)YMIN=P(I,2)
    1 CONTINUE
  101 CONTINUE
C
C     NOW THE OUTER LIMITS ARE SET,
C
      DX=(XMAX-XMIN)*. 1
      DY=(YMAX-YMIN)*.1
      P(201,1)=XMIN-DX
      P(201,2)=YMIN-DY
      P(202,1)=XMAX+DX
      P(2C2,2)=P(201,1)
      P(203,1)=P(202,1)
      P(203,2)=YMAX+DY
      P(204,1)=P(201,1)
      P(2C4,2)=P(203,2)
C
C     NOW THE INITIAL POINT OF REAL DATA IS USEO TO FORM THE FIRST
C     EIGHT SIDES AND FOUR TRIANGLES,
C
      S(1,1)=201
      S(1,2)=202
      S(2,1)=202
      S(2,2)=203
      S(3,1)=203
      S(3,2)=204
      S(4,1)=204
      S(4,2)=201
      S(5,1)=201
      S(5,2)=1
      S(6,1)=202
      S(6,2)=1
      S(7,1)=203
      S(7,2)=1
      S(8,1)=204
      S(8,2)=1
      T(1,1)=1
      T(1,2)=5
      T(1,3)=6
      T(2,1)=6
      T(2,2)=7
      T(2,3)=2
      T(3,1)=3
      T(3,2)=7
      T(3,3)=8
      T(4,1)=4
      T(4,2)=5
      T(4,3)=8
      NS=8
      NT=4
C
C     THIS FILLS IN THE INITIAL VERTICES OF THE TRIANGLES
C
C     NOW PROCEED TO LABEL THE X AND Y COORDIANTES OF TNE TRIANGLES
C
      00 z I=1,4
    2 CALL LOADVER(VERT,P,S,T,I)
```

```
C
C     THE Retaining POINTS ARE NOW ADDED ONE BY ONE
C
      DO 6 I=2,NP
      X=P(I,1)
      Y=P(I,2)
C     X AND Y BECOME THE NEXT POINT TO BE ADDED TO THE MESH
C     NOW PROCEEO TC FINDOUTWHAT TRIANGLE X AND Y ARE IN
C
      IFLAG=0
      CALL INSIDE(VERT,X,Y,NT,ITRI)
C
C     ITRI IS NOW THE TRIANGLE THAT CONTAINS THE POINT I.
C
      IF(ITRI.EQ.-2) GO TO 6
      IF(ITRI.NE.0) GO TO 102
      WRITE(6,200)I
  200 FORMAT(*0*,*POINT*,I5,*  IS NOT IN DOMAIN*)
      GO TO 6
  102 CONTINUE
C
C     HERE, THE NEWTRIANGLES AND SEGMENTS ARE ADOED.
C     WEBEGIN BY LABELINGTHE SIDES ANO VERTICES OF THE FIRST TRIANGLE,
C
      IS1=T(ITRI,1)
      IS2=T(ITRI,2)
      IS3=T(ITRI,3)
      IP1=S(IS2,2)
      IF(S(IS2,1).NE.S(IS1,1).AND.S(IS2,1).NE.S(IS1,2))IP1=S(IS2,1)
      IP2=S(IS1,2)
      IF(S(IS1,1).NE.S(IS2,1).AND.S(IS1,1).NE.S(IS2,2) )IP2=S(IS1,1)
      IP3=S(IS1,2)
      IF(IP3.EQ.IP2)IP3=S (IS1,1)
C
C     NOWTHE NEW LINE SEGMENTS ARE GENERATED.
C
      S(NS+1,1)=IP1
      S(NS+1,2)=I
      S(NS+2,1)=IP2
      S(NS+2,2)=I
      S(NS+3,1)=IP3
      S(NS+3,2)=I
C
C      NOWTHENEW TRIANGLES ARE CREATEO
C
      T(ITRI,1)=IS1
      T(ITRI,2)=NS+2
      T(ITRI,3)=NS+3
      T(NT+1,1)=IS2
      T(NT+1,2)=NS+1
      T(NT+1,3)=NS+3
      T(NT+2,1)=IS3
      T(NT+2,2)=NS+1
      T(NT+2,3)=NS+2
C
C     NOW THE VERTICES OF THE NEW TRIANGLES ARE LOCATED.
C
      CALL LOADVER(VERT,P,S,T,ITRI)
      II=NT+1
      CALL LOADVER(VERT,P,S,T,II)
      II=NT+2
      CALL LOADVER(VERT,P,S,T,II)
C
C     NOW THE, NUMBEROF TRIANGLES AND THE NUMBER OF LINES AREUPDATED.
C
      NT=NT+2
      NS=NS+3
C
```

```
C      NOW THE TRIANGLES THAT SHARE SIDES,IS1,IS2, AND IS3 ARE EACH
C      CHECKED FOR POTENTIAL REORIENTATION,
C
       IS(1)=IS1
       IS(2)=IS2
       IS(3)=IS3
       DO 5 J=1,3
C       HERE TOFINDTWO NEIGHBORING TRIANGLES.
C
       CALL NEIG(IS(J),T,JT1,JT2,NT)
C
C      NOW TO ORDER THE PAIR OF TRIANGLES.
C
       IF(JT2.EQ.0) GO TO 5
       JS1=IS(J)
       JP1=S(JS1,1)
       JP2=S(JS1,2)
C
       DO 3 K=1,3
       JST1=T(JT1,K)
       JST2=T(JT2,K)
       IF(S(JST1,1).EQ.JP1.AND.S (JST1,2).NE. JP2) JS2=JST1
       IF(S(JST1,2) .EQ.JP1.AND.S(JST1 ,1).NE.JP2) JS2=JST1
       IF(S(JST1,1) .EQ.JP2.AND.S(JST1,2).NE.JP1) JS3=JST1
       IF(S(JST1,2).EQ.JP2.AND.S(JST1,1).NE.JP1) JS3=JST1
       IF(S(JST2,1).EQ.JP1.AND.S(JST2,2).NE.JP2) JS4=JST2
       IF(S(JST2,2).EQ.JP1.AND.S(JST2,1).NE.JP2) JS4=JST2
       IF(S(JST2,1).EQ.JP2.AND.S(JST2,2).NE.JP1) JS5=JST2
       IF(S(JST2,2).EQ.JP2.AND.S(JST2,1).NE.JP1) JS5=JST2
     3 CONTINUE
C
       JP3=S(JS3,1)
       IF(JP3.EQ.JP2)JP3=S(JS3,2)
       JP4=S(JS5,1)
       IF(JP4.EQ.JP2)JP4=S(JS5,2)
C
C
C
C       THE TRIANGLES ARE NOW CHECKED TO SEE IF THEY FORM
C      A CONVEX REGICN.
C
       X1=P(JP4,1)
       Y1=P(JP4,2)
       X2=P(JP3,1)
       Y2=P(JP3,2)
       X3=P(JP1,1)
       Y3=P(JP1,2)
       X=P(JP2,1)
       Y=P(JP2,2)
       CALL INNER(X1,Y1,X2,Y2,X3,Y3,X,Y,IC)
       IF(IC.EQ.1) GO TO 5
       X3=X
       Y3=Y
       X=P(JP1,1)
       Y=P(JP1,2)
       CALL INNER(X1,Y1,X2,Y2,X3,Y3,X,Y,IC)
       IF(IC.EQ.1) GO TO 5
C
```

```
C     NOW CHECK FOR BOUNDARY SEGMENTS
C
      IF(JP3.GT.200.OR.JP4.GT.200) GO TO 5
      IF(JP1.GT.200.OR.JP2.GT.200) GO TO 4
      DO 307 KJ=1,NBV
      K1=IB(KJ)
      IF(JP1.NE.K1.AND.JP2.NE.K1) GO TO 307
      IMORE=KJ+1
      ILESS=KJ-1
      IF(ILESS.EQ.0) GO TO 331
      ILESS=IB(ILESS)
      IF(JP1.EQ.ILESS.OR.JP2.EQ.ILESS) GO TO 5
  331 CONTINUE
      IF(IMORE.GT.NBV) GO TO 332
      IMORE=IB(IMORE)
      IF(JP1.EQ.IMORE.OR.JP2.EQ.IMORE) GO TO 5
      GO TO 308
  332 CONTINUE
  307 CONTINUE
  308 DO 309 KJ=1,NBV
      K1=IB(KJ)
      IF(JP3.NE.K1. AND.JP4.NE.K1) GO TO 309
      IMORE=KJ+1
      ILESS=KJ-1
      IF(ILESS.EQ.0) GO TO 431
      ILESS=IB(ILESS)
      IF(JP3.EQ.ILESS.OR.JP4.EQ. ILESS) GO TO 4
  431 CONTINUE
      IF(IMORE.GT.NBV) GO TO 432
      IMORE=IB(IMORE)
      IF( JP3.EQ.IMORE.OR.JP4.EQ. IMORE) GO TO 4
      GO TO 311
  432 CONTINUE

C     IN THIS SECTION, THE FLAGGED TRIANGLES ARE ELIMINATED AND THE
C     LIST COMPACTED.  THE ACTUAL NUMBER OF TRIANGLES REMAINING IS
C     RETURNED AS NT.
C
      T(NT+1,1)=0
      ITOP=NT+1
      IBEGIN=1
   17 DO  10  I=IBEGIN,ITOP
      IF(T(I,1).NE.0) GO TO 10
      GO TO 16
   10 CONTINUE
   16 NEXT=I+1
      DO 9 J=NEXT,ITOP
    9 IF(T(J,1).NE.0) GO TO 13
      GO TO 18
   13 INTV=J-I
      ITOP=ITOP-INTV
      DO 14 K=I,ITOP
      L=K+INTV
      DO 15 M=1,3
   15 T(K,M)=T(L,M)
   14 CONTINUE
      IBEGIN=I+1
      GO TO 17
   18 CONTINUE
      DO 11 I=1,NT
      IF(T(I,1).EQ.0) GO TO 12
   11 CONTINUE
   12 NT=I-1
      RETURN
      END
```

```
      SUBROUTINE LOADVER(VERT,P,S,T,I)
      INTEGER S(607,2),T(403,3)
      INTEGER P1,P2,P3,S1,S2
      DIMENSION VERT(402,6),P(204,2)
      S1=T(I,1)
      S2=T(I,2)
      P1=S(S1,1)
      P2=S(S1,2)
      P3=S(S2,1)
      IF(P3.EQ.P1.OR.P3.EQ. P2) P3=S(S2,2)
      VERT(I,1)=P(P1,1)
      VERT(I,2)=P(P2,1)
      VERT(I,3)=P(P3,1)
      VERT(I,4)=P(P1,2)
      VERT(I,5)=P(P2,2)
      VERT(I,6)=P(P3,2)
      RETURN
      END
```

```
      SUBROUTINE INNER(X1,Y1,X2,Y2,X3,Y3, X,Y,IC)
      DIMENSION A(3,3)
      CALL LOAD(A,X1, Y1,X2,Y2,X3,Y3)
      CALL TRIAREA(A,AREA)
      IF(AREA.EQ.0.) GO TO 8
      CALL LOAD(A,X,Y,X2,Y2,X3,Y3)
      CALL TRIAREA(A,AREA1)
      CALL LOAD(A,X1,Y1,X,Y,X3,Y3)
      CALL TRIAREA(A,AREA2)
      AREA1=AREA1/AREA
      IF(AREA1.LT.0.) GO TO 10
      IF(AREA1.EQ.1) GO TO 9
      AREA2=AREA2/AREA
      IF(AREA2.LT.0.) GO TO 10
      IF(AREA2.EQ.1)GO TO 9
      AREA3=1.-AREA1-AREA2
      IF(AREA3.LT.0.) GO TO 10
      IF(AREA3.EQ.1) GO TO 9
      IC=1
      RETURN
   10 IC=0
      RETURN
    9 CONTINUE
      IF(X1.EQ.X) GO TO 11
      IF(X2.EQ.X) GO TO 12
      IF(X3.EQ.X) GO TO 13
      GO TO 10
   11 IF(Y1.EQ.Y) GO TO 14
      GO TO 10
   12 IF(Y2.EQ.Y) GO TO 14
      GO TO 10
   13 IF(Y3.EQ.Y) GO TO 14
      GO TO 10
   14 IC=-1
      RETURN
    8 CONTINUE
      IC=2
      RETURN
      END
```

```
      SUBROUTINE INSIDE(VERT,X,Y,NT,IPTRI)
      DIMENSICN VERT(402,6)
      DO 1 I=1,NT
      IF(AMIN1(VERT(I,1),VERT(I,2),VERT(I,3)).GT.X) GO TO 1
      IF(AMAX1(VERT(I,1),VERT(I,2),VERT(I,3)).LT.X) GO TO 1
      IF(AMIN1(VERT(I,4),VERT(I,5),VERT(I,6)).GT.Y) GO TO 1
       IF(AMAX1(VERT(I,4),VERT(I,5),VERT(I,6)).LT.Y) GO TO 1
      CALL INNER(VERT(I,1),VERT(I,4),VERT(I,2),VERT(I,5),VERT(I,3),VERT(
     vI,6),X,Y,IC)
      IF(IC.EQ.1) GO TO 2
      IF(IC.EQ.-1) GO TO 3
      IF(IC.EQ.2) Go TO 4
    1 CONTINUE
      WRITE(6,5)X,Y
    5 FORMAT(*0*,*POINT*,2F8.2,2X,*NOT IN ANY TRIANGLE*)
      IPTRI=0
      RETURN
    2 IPTRI=I
      RETURN
    3 IPTRI=-2
      WRITE(6,10)X,Y,I
   10 FORMAT(*0*,*WE HAVE A DUPLICATE POINT*,2F8.2,I5)
      RETURN
    4 IPTRI=I
      RETURN
      END
```

```
      SUBROUTINE LOAD(A,X1,Y1,X2,Y2,X3,Y3)
      DIMENSION A(3,3)
      A(1,1)=X1
      A(1,2)=Y1
      A(1,3)=1.
      A(2,1)=X2
      A(2,2)=Y2
      A(2,3)=1.
      A(3,1)=X3
      A(3,2)=Y3
      A(3,3)=1.
       RETURN
       ENO
      SUBROUTINE CHKA(ITRI,VERT,IFLAG)
      DIMENSICN VERT(402,6),A(3,3)
      X1=VERT(ITRI,1)
      X2=VERT(ITRI,2)
      X3=VERT(ITRI,3)
      Y1=VERT(ITRI,4)
      Y2=VERT(ITRI,5)
      Y3=VERT(ITRI,6)
      CALL LOAD(A,X1,Y1,X2,Y2,X3,Y3)
      CALL TRIAREA(A,AREA)
      IF(AREA.EQ.0.) GO TO 1
      IFLAG=0
      RETURN
    1 IFLAG=1
      RETURN
      END
```

```
      SUBROUTINE SWEEP(P,S,T,NT,ITIME,NIBP)
C
C     THIS SUBROUTINE SWEEPS THROUGH THE TRIANGLES AND COMPARES
C          NEIGHBORS FOR GOODNESS.
C
C     P IS THE POINT ARRAY AS DESCRIBED IN TRIAN.
C     S IS THE SIDE ARRAY AS DESCRIBED IN TRIAN.
C     T IS THE TRIANGLE ARRAY AS DESCRIBED IN TRIAN.
C     ITIME IS A COUNTER TELLING US HOW MANY TRIANGLES HAVE BEEN CHANGED
C     NIBP IS THE NUMBER OF INTERIOR BOUNDARY POINTS*
C
      INTEGER S(607,2),T(403,3)
      COMMON/BOUND/IB(75),BV(75),NBV
      DIMENSION P(204,2)
      ITIME=0
C
C     HERE THE SWEEP BEGINS,
C
      DO 100 I=1,NT
      DO 2 J=1,3
C
C     BEGIN BY EXAMINING THE J SIDE OF TRIANGLE I
C
      IS=T(I,J)
C
C     HERE WE LOCATE THE TWO NEIGHBORING TRIANGLES THAT HAVE SIDE IS
C
      CALL NEIG(IS,T,JT1,JT2,NT)
      IF(JT2.EQ.0) GO TO 2
C
C     PROCEED TO CODE THE POINTS OF THE NEIGHBORING TRIANGLES.
C
      JP1=S(IS,1)
      JP2=S(IS,2)
      IS1=T(JT1,1)
      IF(IS.EQ.IS1)IS1=T(JT1,2)
      JP3=S(IS1,1)
      IF(JP3.EQ.JP2.OR.JP3.EQ.JP1) JP3=S(IS1,2)
      IS1=T(JT2,1)
      IF(IS.EQ.IS1)IS1=T(JT2,2)
      JP4=S(IS1,1)
      IF(JP4.EQ.JP2.OR.JP4.EQ.JP1) JP4=S(IS1,2)
C
C     NOW CHECK FOR CONVEX TRIANGLES*
C
      X1=P(JP4,1)
      Y1=P(JP4,2)
      X2=P(JP3,1)
      Y2=P(JP3,2)
      X3=P(JP1,1)
      Y3=P(JP1,2)
      X=P(JP2,1)
      Y=P(JP2,2)
```

```
C
      CALL INNER(X1,Y1,X2,Y2,X3,Y3,X,Y,IC)
      IF(IC.EQ.1) GO TO 2
      X3=X
      Y3=Y
      X=P(JP1,1)
      Y=P(JP1,2)
C
      CALL INNER(X1,Y1,X2,Y2,X3,Y3,X,Y,IC)
      IF(IC.EQ.1) GO TO 2
C
C
C     NOW CHECK FOR BOUNDPRY SEGMENTS
C
      IF(JP3.GT.200.OR.JP4.GT.200) GO TO 2
      IF(JP1oGTo2CDSOR9JP2.GTO200) GO TO 4
C
      DO 307 KJ=1,NBV
      K1=IB(KJ)
      IF(JP1.NE.K1.AND.JP2.NE.K1) GO TO 307
      ILESS=KJ-1
      IMORE=KJ+1
      IF(ILESS.EQ.0) GO TO 331
      ILESS=IB(ILESS)
      IF(JP1o EQ,ILESS,C?,JP2.EO,ILESS) GO TO 2
  331 CONTINUE
      IF(IMCRE.GT.NBV) GO TC 332
      IMORE=IB(IMORE)
      IF(JP1.EQ.IMORE.OR.JP2.EQ.IMORE) GO TO 2
      GOTO 308.
  332 CONTINUE
  307 CONTINUE
  308 DO 309 KJ=1,NBV
      K1=IB(KJ)
      IF(JP3.NE,K1oANDoJP4.NE.K1) GO TO 309
      ILESS=KJ-1
      IMORE=KJ+1
      IF(ILESS.EQ.0) GO .10 431
      ILESS=IB(ILESS)
      IF(JP3.EQ.ILESS.OR.JP4.EQ.ILESS) GO TO 4
  431 CONTINUE
      IF(IMORE.GT.NBV) GO TO 432
      IMORE=IB(IMORE)
      IF(JP3.EQ.IMORE.OR.JP4.EQ.IMORE) GO TO 4
      Go TO 311
  432 CONTINUE
  309 CONTINUE
  311 CONTINUE
C
C     NOW PROCEEO TC CALCULATE ANO COHPARE THE GOODNESS OF THE
C          NEIGHBORING TRIANGLES.
C
      G1=GOOD(JP1,JP2,JP3,P)
```

```
      G2=GOOD(JP1,JP2,JP4,P)
      G3=GOOD(JP1,JP3,JP4,P)
      G4=GOOD(JP2,JP3,JP4,P)
      GW=AMIN1(G3,G4)
      IF(G1.LT.GW) GO TO 4
      IF(G2.LT.GW)GO TO 4
      GO TO 2
    4 CONTINUE
C
C     THIS IS WHERE THE SWITCHING OF THE TRIANGLES ARE DONE.
C          WE BEGIN BY LABELING THE SIDES FOR IDENTIFICATION.
C
      JS1=IS
      S(JS1,1)=JP4
      S(JS1,2)=JP3
      DO 200 K=1,3
      JST1=T(JT1,K)
      JST2=T(JT2,K)
      IF(S(JST1,1).EQ.JP1.AND.S(JST1,2).NE.JP2) JS2=JST1
      IF(S(JST1,2).EQ.JP1.AND.S(JST1,1).NE.JP2) JS2=JST1
      IF(S(JST1,1).EQ.JP2.AND.S(JST1,2).NE.JP1) JS3=JST1
      IF(S(JST1,2).EQ.JP2.AND.S(JST1,1).NE.JP1) JS3=JST1
      IF(S(JST2,1).EQ.JP1.AND.S(JST2,2).NE.JP2) JS4=JST2
      IF(S(JST2,2).EQ.JP1.AND.S(JST2,1).NE.JP2) JS4=JST2
      IF(S(JST2,1).EQ.JP2.AND.S(JST2,2).NE.JP1) JS5=JST2
      IF(S(JST2,2).EQ.JP2.AND.S(JST2,1).NE.JP1) JS5=JST2
  200 CONTINUE
C
C
C     NOW THE TRIANGLES ARE ALTERED.
C          AND THE CHANGED RECORDED.
C
      T(JT1,1)=JS1
      T(JT1,2)=JS2
      T(JT1,3)=JS4
      T(JT2,1)=JS1
      T(JT2,2)=JS3
      T(JT2,3)=JS5
      ITIME=ITIME+1
      GO TO 100
    2 CONTINUE
  100 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE FINDBP(IB,IP,IBTRI,NTRI,NBV)
      DIMENSION IB(1),IP(3,1),IBTRI(1)
C     THIS SUBROUTINE CALCULATES THE NUMBER OF POINTS IN EACH TRIANGLE
C     AND ORDERS THE BOUNDARY POINTS.
      DO 1601 MN=1,NTRI
 1601 IBTRI(MN)=0
      DO 622 K=1,NBV
      DO 622 J=1,NTRI

      DO 621 I=1,3
      IF(IB(K).NE.IP(I,J)) GO TO 621
      IBTRI(J)=IBTRI(J)+1
      I2=IBTRI(J)
      NUM=IP(I2,J)
      IP(I2,J)=IP(I,J)
      IP(I,J)=NUM
      GO TO 622
  621 CONTINUE
  622 CONTINUE
C
C
```

```
C     THIS SECTION CHECKS TO SEE IF THE BOUNDARY TRIANGLES AND NUMBERING
C     SYSTEM ARE CONSISTENT
C
      DO 633 J=1,NTRI
      NUM=IBTRI(J)-2
      IF(NUM)633,634,643
  634 DO 637 K=1,NBV
      IF(IB(K).NE.IP(1,J)) GO TO 637
      IJ=K+1
      IF(IB(IJ).EQ.IP(2,J)) GO TO 633
      IF(IP(1,J).EQ.IB(1).AND.IP(2,J).EQ.IB(NBV))GO TO 641
      DO 734 M=IJ,NBV
      IF(IB(M).EQ.IP(2,J)) GO TO 758
      GO TO 734
  758 IBTRI(J)=0
      GO TO 633
  734 CONTINUE
      DO 736 M=1,NBV
      IF(IB(M).EQ.IP(3,J)) GO TO 640
  736 CONTINUE
      GO TO 635
  641 CONTINUE
      NUM=IP(1,J)
      IP(1,J)=IP(2,J)
      IP(2,J)=NUM
      GO TO 633
  637 CONTINUE
  635 IBTRI(J)=5
C
C     IBTRI(J)=5INDICATES THAT NO BOUNDARY POINTS WERE FOUND INTHE
C     TRIANGLE EVEN THOUGH IBTRI HAS OVER 2 ORIGINALLY
C
```

```
      FUNCTION GOOD(JP1,JP2,JP3,P)
      DIMENSION AL(3),P(204,2)
      DIMENSION X(2),Y(2)
      X(1)=P(JP1,1)
      X(2)=P(JP2,1)
      Y(1)=P(JP1,2)
      Y(2)=P(JP2,2)
      AL(1)=ALENGTH(X,Y)
      X(2)=P(JP3,1)
      Y(2)=P(JP3,2)
      AL(2)=ALENGTH(X,Y)
      X(1)=P(JP2,1)
      Y(1)=P(JP2,2)
      AL(3)=ALENGTH(X,Y)
      ALM=AL(1)
      I=1
      DO 2 J=2,3
      IF(AL(J).GT.ALM) GO TO 1
      GO TO 2
    1 I=J
      ALM=AL(J)
    2 CONTINUE
      RLL=0.
      DO 3 J=1,3
      IF(J.NE.I)RLL=RLL+AL(J)
    3 CONTINUE
      GOOD=RLL/ALM
      RETURN
      END
```

```
      SUBROUTINE ELIM(IBTRI,IP,X,Y,NTRI)
      DIMENSICN IBTRI(1) ,IP(3,1),X(1),Y(1 )
      COMMON/IA/A (3,3)
      COMMON/IB/B(3)
      COMMON/INT/ISUM(300),IPROD(300)
C
C     THIS SUBROUTINE ELIMINATES TRIANGLES OUTSIDE OF THE DOMAIN
C
      DO 662 J=1,NTRI
      IF(IBTRI(J).LT.2) Go TO 662
  663 NN=IP(1,J)
      M=IP(2,J)
      L=IP(3,J)
  664 DX=X(M)-X(NN)
      DY=Y(M)-Y(NN)
      DS=((DX**2.)+(DY**2.))**.5
      DCOSX=DY/DS
      DCOSY=-DX/DS
  666 A(1,1)=X(L)
      A(1,2)=Y(L)
      A(1,3)=1.
      A(2,1)=X(NN)
      A(2,2)=Y(NN)
      A(2,3)=1.
      A(3,1)=X(M)
      A(3,2)=Y(M)
      A(3,3)=1.
      B(1)=1.
      B(2)=0.
      B(3)=0.
      CALL SOLVE(A,B)
 "667 DX=X(NN)+0.5*DX+0.01*DS*DCOSX
      DY=Y(NN)+0.5*DY+0.1*DS*DCOSY
      DS=B(1)*DX+B(2)*DY+B(3)
      IF(DS.LT.0) GO TO 662
      IBTRI(J)=6
  662 CONTINUE
      I=NTRI
  262 DO 674 J=1,NTRI
      IF(IBTRI(J).NE.6) GO TO 674
  272 IF(J.EQ.I) GO TO 677
      I=I-1
      DO 676 K=J,I
      L=K+1
      IBTRI(K)=IBTRI(L)
      IP(1,K)=IP(1,L)
      IP(2,K)=IP(2,L)
  676 IP(3,K)=IP(3,L)
      IBTRI(NTRI)=0
      IF(IBTRI(J).EQ.6) GO TO 272
  674 CONTINUE
  677 NTRI=I+1
      DO 802 I=1,NTRI


      GO TO 633
  643 00 649 K=1,NBV
      IF(IB(K).NE.IP(1,J)) GO TO 649
      IJ=K+1
      IF(IB(IJ).EQ.IP(2,J)) Go To 651
      KK=NBV-1
      IF(IP(1,J).EQ.IB(1).AND.IP(2,J).EQ.IB(KK)) GO TO 735
      00 737 M=IJ,NBV
  737 IF(IB(M).EQ.IP(2,J)) GO TO 651
      GO TO 635
```

```
  735 CONTINUE
      IJ=NBV-1
      IF(IB(IJ).NE.IP(2,J)) GO TO 640
      IF(IB(NBV).NE.IP(3,J)) GO TO 640
      NUM=IP(1,J)
      IP(1,J)=IP(2,J)
      IP(2,J)=IP(3,J)
      IP(3,J)=NUM
  649 CONTINUE
      GO TO 633
  651 IJ=K+2
      IF(IB(1).EQ.IP(3,J)) GO TO 633
      IF(IB(IJ).EQ.IP(3,J)) GO TO 633
      IF(IP(1,J).EQ.IB(1).AND.IP(3,J).EQ.IB(NBV)) GO TO 739
      DO 852 M=IJ,NBV
  852 IF(IB(M).EQ.IP(3,J)) GO TO 743
      GOTO 635
  743 IBTRI(J)=3
      GOTO 633
  739 IF(IB(NBV).NE.IP(3,J)) GO TO 640
      NUM=IP(1,J)
      IP(1,J)=IP(3,J)
      IP(3,J)=IP(2,J)
      IP(2,J)=NUM
      GO TO 633
  640 IBTRI(J)=4
C
  633 CONTINUE
C
C     IBTRI(J)=4INDICATES THAT THE BOUNDARY POINTS WERE NOT IN SEQUENCE
      RETURN
      END
```

```
      SUBROUTINE BYPASS(I,IYES)
      COMMON/BOUND/IB(75),BV(75),NBV
      COMMON/CUTOFF/NIBP,NFLX,JJ1,JJ2,DEEP
      COMMON/IDEPTH/DEPTH(200)
      IYES=0
      J=IB(I)
      IF(DEPTH(J).LE.DEEP)IYES=1
      NF=NFLX+NIBP
      IF(I.LE.NF) IYES=1
      RETURN
      END
```

```
      ISUM(I)=IP(1,J)+IP(2,J)+IP(3,J)
  802 IPROD(I)=IP(1,J)*IP(2,J) ● IP(31J)
      ICOUNT=0
  807 IEND=NTRI-ICOUNT
      DO 804 I=1,IEND
      IF(ISUM(I).NE.ISUM(IEND)) GO TO 804
      IF(IPROD(I).NE.IPROD(IEND) ) GO TO 804
      IF(I.EQ.IEND) CO TO 804
      ICOUNT=ICCUNT+1
      GO TO 807
  804 CONTINUE
      NTRI=NTRI-ICOUNT
      RETURN
      END
```

```
      SUBROUTINE WHCUT(DST,X1,Y1,EDST)
      COMMCN/BOUND/IB(75),BV(75),NBV
      COMMON/IX/X(200)
      COMMON/IY/Y(200)
      DIMENSION DST(1)
      CALL WHERE(X1,Y1,J1,J2)
      I2=IB(J1)
      ADD=((X(I2)-X1)**2.+(Y(I2)-Y1)**2.)**.5
      EDST=DST(J1)+ADD
      RETURN
      ENO

      SUBROUTINE WHIN(DST)
      COMMON/BOUND/IB(75),BV(75),NBV
      CCMMON/IX/X(200)
      COMMON/IY/Y(200)
      DIMENSICN DST(1)
      DST(1)=0.
      00 9 I=2,NBV
      J=IB(I)
      L=I-1
      K=IB(L)
      DT=((X(K)-X(J))**2.+(Y(K)-Y(J))**2.)**.5
    9 DST(I)=DST(L)+DT
      RETURN
      END

      SUBROUTINE  GUESS(X1SY1OJ1,JZ*ELEV)
      COMMON/BOUND/IB(75),BV(75),NBV
      CCMMON/IX/X(200)
      COMMON/IY/Y(200)
      COMMCN/NUMB/NVRTX,NTRI,LIST,IPUNCH,NEWBV
      I1=IB(J1)
      I2=IB(J2)
      DS1=((X(I1)-X1)**2.+(Y(I1)-Y1)**2.)**.5
      DS2=((X(I2)-X1)**2.+(Y(I2)-Y1)**2.)**.5
      DS=DS1+DS2
      ELEV=(DS2/DS)*BV(J1)+(DS1/DS)*BV(J2)
      RETURN
      END

      SUBROUTINE ALTER(XI,YI)
C
C          THIS RCUTINEALTERSTHE OLD BOUNDARY CONDITONS.
C
      COMMON/IRHS/DST(100),EI(100)
      COMMCN/IDEPTH/DEPTH(200)
      COMMON/BOUND/IB(75),BV(75),NBV
      COMMON/IX/X(200)
      COMMON/IY/Y(200)
      COMMON/NUMB/NVRTX,NTRI,LIST,IPUNCH,NEWBV
      COMMON/CUTCFF/NIBP,NFLX,J1,J2
      DIMENSION XI(1),YI(1)
      DO 10 I=1,NBV
      CALL BYPASS(I,IYES)
      IF(IYES.EQ.1) GO TO 10
    9 DH=EI(I)-BV(I)
      CALL WHCUT(DST,XI(I),YI(I),EDST)
      IF(EDST.LT.DST(I)) GO TO 10
    8 CALL WHERE(XI(I),YI(I),J1,J2)
      CALL GUESS(XI(I),YI(I),J1,J2,ELEV)
      BV(I)=ELEV-DH
   10 CONTINUE
      RETURN
      END
```

```
      Subroutine BTW(X1,X2,Y1,Y2,X,Y,IYES)
      IF(AMAX1(X1,X2).LT.X) GO TO 1
      IF(AMIN1(X1,X2).GT.X) GO TO 1
      IF(AMAX1(Y1,Y2).LT.Y) GO TO 1
      IF(AMIN1(Y1,Y2).GT.Y) GO TO 1
      IYES=1
      RETURN
    1 IYES=0
      RETURN
      ENO
```

```
      SUBROUTINE WHERE(X1,Y1,J1,J2)
      COMMON/IGRAD/IFLG,IBEGIN,IEND,NIBP,NH
      COMMON/IX/X(200)
      COMMON/IY/Y(200)
      COMMON/BOUND/IB(75),BV(75),NBV
      COMMON/NUMB/NVRTX,NTRI, LIST,IPUNCH,NEWBV
      I=NIBP+1
      I1=IB(I)
      I=IB(NBV)
      CALL BTW(X(I) ,X(I1),Y(I),Y(I1) ,X1,Y1,IYES)
      IF(IYES.NE.1) GO TO 1
      J1=NBV
      J2=NIBP+1
      RETURN
    1 ISTOP=NBV-1
      00 2 I=1,ISTOP
      J=I+1
      K1=IB(I)
      K2=IB(J)
      CALL BTW(X(K1),X(K2),Y(K1),Y(K2) ,X1,Y1,IYES)
      IF(IYES.EQ.0.) GO TO 2
      J1=I
      J2=J
      RETURN
    2 CONTINUE
      WRITE(6,10)X1,Y1
   10 FORMAT(*0*,2F10.2,3X,*NOT ON BOUNDARY*)
      RETURN
      END
```

```
      SUBROUTINE INTG(X1,Y1,X2,Y2,ELEV1,ELEV2, ITRI,SA,DP)
      DIMENSION SA(1)
      COMMON/IALPHA/ALPHA(200)
      COMMON/WIND/TAUX,TAUY,CURL
      COMMON/ICCNST/CONST1,CONST2
      COMMON/IIP/IP(3,350)
      IF(X1.EQ.X2.AND.Y1.EQ.Y2) GO TO 1
      DS=((X1-X2) **2.+(Y1-Y2) **2.)**.5
      C A L L  JACOB(ITRI, DDX,DDY,AJ,SA)
      CALL DDN(X1,Y1,X2,Y2,DDX,DDY,DN)
      IF(ABS(DN).LT..000001) GO TO 2
      ELEV2=(CONST1*AJ+CURL)*DS/DN+ELEV1
      RETURN
    1 ELEV2=0.
      RETURN
    2 ELEV2=ELEV1
      WRITE(6,10) ITRI
   10 FORMAT(1H0,*WE HAVE RUN INTO A TRIANGLE WITH NO DEPTHGRADIENTS*,2
     CX,I3)
      RETURN
      END
```

```
      SUBROUTINE SOLVER(ELEV1,X1,Y1,ELEV2,X2,Y2,IGLB,SA)
C
C          THIS  IS  THE ROUITNE THAT SOLVES THE FIRST ORDER EQUATION.
C
      COMMON/CUTOFF/IFINIS,NFLX,J1,J2
      COMMON/IGRAD/IFLG,IBEGIN,IEND, NIBP,IDONE
      COMMON/IDEPTH/DEPTH(200)
      COMMON/NUMB/NVRTX,NTRI,NOINTG,IPINTG,NEWBV
      DIMENSION SA(1)
      DP=DEPTH(IGLB)
      IFINIS=0
      IEND=0
      IBEGIN=0
      IFLG=0
      IDONE=0
C
C          BEGIN BYLOCATING THE FIRST TRIANGLE WE WILL WORK WITH.
C
      CALL FIRST(IGLB,ITRI,0)
      IF(ITRI.NE.0) GO TO 1
      X2=X1
      Y2=Y1
      ELEV2=ELEV1
      RETURN
C
C          NOW FIND THEPOINT ALONG THE TRIANGLE BOUNDARY WHICH HAS THE
C     SAME DEPTH  A S  THE POINT WE CAME FROM.
C
    1 CALL FNDPT(X1 ,Y1,ITRI,X2,Y2,DP)
C
C          NOW INTEGRATE FROM THE ORIGINAL POINT TO THE SECONO POINT
C     WITHIN THE TRIANGLE.
C
      CALL INTG(X1,Y1,X2,Y2,ELEV1,ELEV2,ITRI,SA,DP)
      ELEV1=ELEV2
      IENC=IEND+1
      IF(IEND.EQ.NTRI) GO  TO 7
C
C          NOW LOCATE THE NEXT TRIANGLE THE DEPTH CONTOUR ENTERS,
C     THEN GO BACK TO ONE AND FIND THE SECOND POINT AGAIN.
C
      CALL EXTEND(X2,Y2,ITRI,NEWTRI)
      IF(NEWTRI.EQ.0) RETURN
    5 X1=X2
      Y1=Y2
      ITRI=NEWTRI
      GO TO 1
C
    7 WRITE(6,100)DP
  100 FORMAT(1H0,*WE ARE LOST FOLLOWING*,1X,F6.2,1X,*DEPTH CONTOUR-)
      RETURN
      END
```

```
      SUBROUTINE BETW(VAL,VAL1,VAL2,IYES)
      COMMCN/IGRAD/IFLG,IBEGIN,IEND,NN,NM
      IFLG=0
      IYES=0
      IF(VAL.LE.VAL1.AND.VAL.GE.VAL2) CO TO 1
      IF(VAL.GE.VAL1.AND.VAL.LE.VAL2) GO TO 1
      RETURN
    1 IYES=1
      IF(VAL.EQ.VAL1)IFLG=1
      IF(VAL.EQ.VAL2) IFLG=1
       RETURN
      ENO
```

```
      SUBROUTINE FIRST(IGLB,ITRI,IOLD)
      COMMON/CUTOFF/IFINIS,NFLX,IP1 ,IP2
      COMMON/IGRAD/IFLG,IBEGIN,IEND,NIBP,IDONE
      COMMON/IIP/IP(3,350)
      COMMON/IX/X(200)
      COMMON/IY/Y(200)
      COMMON/IDEPTH/DEPTH(200)
      COMMON/NUMB/NVRTX,NTRI, LIST,IPUNCH,NEWBV
      DC 2 I=1,NTRI
      IF(I.EQ.ICLD) GO TO 2
      IF(I.EQ.IFINIS) GO TO 2
      IF(I.EQ.IDONE) GC TO 2
      DO 1 J=1,3
      K=IP(J,I)
      IF(IGLB.EQ.K) GO TO 3
    1 CONTINUE
      GO To 2
    3 CONTINUE
      I1=IP(1,I)
      I2=IP(2,I)
      I3=IP(3,I)
      IF(IGLB.EQ.I1) GC TO  4
      IF(IGLB.EQ.I2) GO TC 5
      IF(IBEGIN.EQ.I1) GO TC  B
      IF(IBEGIN.EQ.I2) GO TC8
      CALL BETW(DEPTH(IGLB),DEPTH(I1),DEPTH(I2),IYES)
      IF(IYES.EQ.1) GO TO 6
      GO    TO 2
    5 CONTINUE
      IF(IBEGIN.EQ.I1) GO TC 8
      IF(IBEGIN.EQ.I3) GO TC 8
      CALL BETH(DEPTH(IGLB),DEPTH(I1),DEPTH(I3),IYES)
      IF(IYES.EQ.1) GO TO 6
      GO TO 2
    4 CONTINUE
      IF(IBEGIN.EQ.I2) GO TO 8
      IF(IBEGIN.EQ.I3) GO TO 8
      CALL BETW(DEPTH(IGLB),DEPTH(I2),DEPTH(I3),IYES)
      IF(IYES.EQ.1) GO TO 6
      GO TO z
    8 IBEGIN=0
    2 CONTINUE
      ITRI=0
      RETURN
    6 ITRI=I
      IF(IFLG.EQ.0) GO TO 7
      IBEGIN=IGLB
      RETURN
    7 IBEGIN=0
      RETURN
      ENO
```

```
      SUBROUTINE VERTCH(X1,Y1,IOLD,IYES,IGLB)
      COMMON/BOUND/IB(75),BV(75),NBV
      COMMON/IIF/IP(3,350)
      COMMON/IX/X(200)
      COMMON/IY/Y(200)
      COMMON/NUMB/NVRTX,NTRI,LIST, IPUNCH,NEWBV
      IF(IOLD.LT.0) IOLD=-IOLD
      DO 1 I=1,3
      K=IP(I,IOLD)
      DX=X1-X(K)
      DY=Y1-Y(K)
      DS=(DX*DX+DY*DY)**.5
      IF(DS.LT..00000001) GO TO 2
    1 CONTINUE
      IYES=0
      RETURN
    2 IYES=1
      DO 3 I=1,NBV
      J=IB(I)
      IF(J.EQ.K) GO TO 4
    3 CONTINUE
      IGLB=K
      RETURN
    4 IGLB=0
      RETURN
      END
```

```
      SUBROUTINE EXTEND(X1,Y1,IOLD,NEWTRI)
      COMMON/IIF/IP(3,350)
      COMMON/IGRAD/IFLG,IBEGIN,IEND,NIBP,IDONE
      COMMON/CUTOFF/IFINIS,NFLX,J1,J2
      COMMON/NUMB/NVRTX,NTRI,LIST,IPUNCH,NEWBV
      CALL VERTCH(X1,Y1,IOLD,IYES,IGLB)
      IF(IYES.EQ.1)GO TO 7
      DO 5 I=1,NTRI
      IF(I.EQ.IOLD) GO TO 5
      DO 4 J=1,2
      K=IP(J,I)
      IF(K.NE.J1.AND.K.NE.J2) GO TO 4
      JJ=J+1
      DO 3 L=JJ,3
      K=IP(L,I)
      IF(K.NE.J1.AND.K.NE.J2) GO TO 3
      NEWTRI=I
      RETURN
    3 CONTINUE
    4 CONTINUE
    5 CONTINUE
    6 NEWTRI=0
      RETURN
    7 IF(IGLB.EQ.0) GO TO 6
C     RECORD WHENCE WE CAME.
      IFINIS=IOLD
      CALL FIRST(IGLB,NEWTRI,IOLD)
C     RECORD TO WHERE WE TREK.
      IDONE=NEWTRI
      RETURN
      END
```

```
      SUBROUTINE DDCN(X1,Y1,X2,Y2,DDX,DDY,DN)
      DX=X2-X1
      DY=Y2-Y1
      DS=(DX*DX+DY*DY)**.5
      DN=DX/DS*DDY-DY/DS*DDX
      RETURN
      END
```

```
      SUBROUTINE JACOB(ITRI ,DDX,DDY,AJ,SA)
      COMMON/IALPHA/ALPHA(200)
      COMMON/IDEPTH/DEPTH(200)
      COMMON/IIP/IP(3,350)
      DIMENSION SA(1)
      DIMENSION ALPH(3)
      COMMON/IA/A(3,3)
      I1=IP(1,ITRI)
      I2=IP(2,ITRI)
      I3=IP(3,ITRI)
      CALL FILL(ITRI,A)
      CALL ALPHX(ITRI,DAX,DAY,ALPHA,SA,ALPH)
      CALL GRAD(DEPTH(I1),DEPTH(I2),DEPTH(I3),DDX,DDY,CC)
      AJ=DAY*DDX-DAX*DDY
      RETURN
      END
```

```
      SUBROUTINE INTERS(DP,X2,Y2,X1,Y1)
      COMMON/IX/X(200)
      COMMON/IY/Y(200)
      COMMON/IIP/IP(3,350)
      COMMON/IDEPTH/DEPTH(200 )
      COMMON/CUTOFF/NIBP,NFLX,IP1,IP2
      D1=DEPTH(IP1)
      D2=DEPTH(IP2)
      DS1=DP-D1
      DS2=D2-DP
      DS=D2-D1
      W1=DS2/DS
      W2=DS1/DS
      IF(D1.NE.D2) GO TO 15
      X2=X(IP2)
      Y2=Y(IP2)
      IF(X2.NE.X1.AND.Y2.NE.Y1) GO TO 20
      X2=X(IP1)
      Y2=Y(IP1)
   20 RETURN
   15 CONTINUE
      X2=X(IP1)*W1+X(IP2)*W2
      Y2=Y(IP1)*W1+Y(IP2)*W2
      RETURN
      END
```

```
      SUBROUTINE FNDPT(X1,Y1,ITRI,X2,Y2,DP)
      COMMON/IX/X(200)
      COMMON/IY/Y(200)
      COMMCN/IIF/IP(3,350)
      COMMON/IDEPTH/DEPTH(200)
      COMMON/NUMB/NVRTX,NTRI,LIST,IPUNCH,NEWBV
      COMMON/CUTOFF/NIBP,NFLX,IP1, IP2
      IFLG=0
      I1=IP(1,ITRI)
      I2=IP(2,ITRI)
      I3=IP(3,ITRI)
      DO 10 I=1,3
      IF(I.EQ.1) GO TO 9
      IF(I.EQ.2) GO TO 8
      IP1=I2
      IP2=I3
      CALL BETH(DP,CEPTH(IP1),DEPTH(IP2) ,IYES)
      IF(IYES.EQ.0) GO TO 10
      GO TO 7
    9 IP1=I1
      IP2=I2
      CALL EETH(DP,CEPTH(IP1),DEPTH(IP2),IYES)
      IF(IYES.EQ.0) GO TO 10
      GO TO 7
    8 IP1=I1
      IP2=I3
      CALL EETH(DP,DEPTH(IP1),DEPTH(IP2)● IYES)
      IF(IYES.EQ.0) GO TO 10
    7 CALL INTERS (DP,X2,Y2,X1,Y1)
      DX=(X1-X2)**2.
      DY=(Y1-Y2)**2.
      IF(DX.LT.1.0E-13.AND.DY.LT. 1oOE-13) GO TO 10
      RETURN
   10 CONTINUE
      WRITE(6,50)ITRI,X1,Y1,DP
   50 FORMAT(*0*,*WE ARE LOST IN TRI.*,I5,2X,*FROM POINT*,2F7.3,2X,*DEPT
     vH*,F6.3)
      IFLG=1000
      RETURN
      ENO
```

```
      SUBROUTINE PPLOT(X,Y,NPTS,XSTRT,XSIZE ,YSIZE,NDIV,NFL,NDEC,YMAX)
      DIMENSION A(120),IX(200),IY(200),SYM(5),FRMT(3),TLAB(3),SB(30)
      DIMENSION X(1),Y(1)
      DATA SYM/1H ,1H*,1H0,1H-,1HX,1HI/
C
C
C     AN ALL PURPOSE PRINTER PLOT ROUTINE WRITTEN IN STANDARDFORTRAN.
C
C     X AND Y ARE COORDINATES OF POINTS TO BE PLOTTED,
C     NPTS ARE NUMBER OF POINTS TO BE PLOTTED.
C     XSIZE IS THE SIZE OF THE PLOT IN INCHES IN THE X DIRECTION.
C     LIKEWISE FOR YSIZE.
C     XSTRT IS THE VALUE CF THE THEMINIMUM X VALUE.
C     NDIV IS THE NUMBER CF PARTITIONS THE X AXIS MILL BE DIVIDED INTO
C     AND LABLED.
C     NFL IS THE FIELD LENGTH OF THE LABEL IN F FORMAT.
C     NDEC IS THE NUMBER OF DECIMAL POINTS THERE WILLBE IN THE FIELD
C     LENGTH.
C     YMAX IS CALCULATE,
C
C
      CALL SCALE(X,Y,IX,IY,NPTS,XSIZE,YSIZE,YPTS,XMIN,XMAX,XPTS,YMAX)
C
      JX=IFIX(XSTRT/.1)
      IYPTS=IFIX(YPTS)
      IXPTS=IFIX(XPTS)
      IXTCT=IXPTS+JX
      NSKP=JX-NFL-1
C
C     TOP LABEL.
C
      NCHR=23
      ENCODE(23,105,TLAB)
  105 FORMAT(23HNEW BOUNDARY ELEVATIONS)
      CALL TLB(TLAB,NCHR,IXPTS,JX)
C
C     SIDELABEL SET IN SUBROUTINE SLB.
      CALL SLB(SB,IS,IE,IYPTS)
C
      ENCODE(21, 100,FRMT)NSKP,NFL, NDEC
  100 FORMAT(5H(1H ,,I2,4HA1,F,I1, 1H.,I1,7H, 110A1})
      DO 6 I=1,JX
    6 A(I)=SYM(1)
      DO 1 I=JX,110
    1 A(I)=SYM(4)
      WRITE(6,5)(A(I),I=1,110)
    5 FORMAT(1H ,110A1)
      IC=1
      DO 3 I=1,IYPTS
      DO 2 IJ=1,110
    2 A(IJ)=SYM(1)
      IF(I.LT.IS.OR.I .GT.IE) GO TO 13
      A(1)=SB(IC)
```

```
      SUBROUTINE TOP(NDIV,XMIN,XMAX,A,NFL,NDEC,ISTRT,NXPTS)
      DIMENSION A(1),FRMT(3)
      NRKS=NDIV+1
      AINT=(XMAX-XMIN)/FLOAT(NDIV)
      DO 1 I=1,NRKS
      J=I-1
    1 A(I)=FLOAT(J)*AINT+XMIN
      RT=FLOAT(ISTRT)-FLOAT(NFL)/2.
      SKP=FLOAT(NXPTS)/FLOAT(NDIV)-FLOAT(NFL)
      IRT=IFIX(RT)
      NSKP=IFIX(SKP)
      ENCODE(22,100,FRMT)IRT,NRKS,NFL,NDEC,NSKP
  100 FORMAT(5H(1H,,I2,2HX,,I2,2H(F,I1,1H.,I1,1H,,I2,3HX)))
      WRITE(6,FRMT) (A(I),I=1,NRKS)
      RETURN
      END


      SUBROUTINE FILL(IT,A)
      DIMENSION A(3,3)
      COMMON/IIP/IP(3,350)
      COMMON/IY/Y(200)
      COMMON/IX/X(200)
      I1=IP(1,IT)
      I2=IP(2,IT)
      I3=IP(3,IT)
      A(1,1)=X(I1)
      A(1,2)=Y(I1)
      A(1,3)=1.
      A(2,1)=X(I2)
      A(2,2)=Y(I2)
      A(2,3)=1.
      A(3,1)=X(I3)
      A(3,2)=Y(I3)
      A(3,3)=1.
      RETURN
      ENO


      SUBROUTINE SCALE(X,Y,IX,IY,NPTS,XSIZE,YSIZE,YPTS,XMIN,XMAX,XPTS,YX
     V)
      DIMENSION X(1),Y(1),IX(1),IY(1)
      XMIN=X(1)
      YMIN=Y(1)
      XMAX=X(1)
      YMAX=Y(1)
      00 2 I=2,NPTS
      XMIN=AMIN1(XMIN,X(I))
      YMIN=AMIN1(YMIN,Y(I))
      XMAX=AMAX1(XMAX,X(I))
      YMAX=AMAX1(YMAX,Y(I))
    2 CONTINUE
      IF(YX.GT.0.)YMAX=YX
      XPTS=XSIZE/.1
      YPTS=YSIZE/.167
      XRANGE=XMAX-XMIN
      YRANGE=YMAX-YMIN
      XRES=XPTS/XRANGE
      YRES=YPTS/YRANGE
      DO 10 I=1,NPTS
      IX(I)=IFIX((X(I)-XMIN)*XRES)
   10 IY(I)=IFIX((Y(I)-YMIN)*YRES)
      RETURN
      ENO
```

```
      IC=IC+1
   13 CONTINUE
      A(JX)=SYM(6)
      IFLG=0
      A(110)=SYM(6)
      00 4 J=1,NPTS
      IF(IY(J).NE.I) GO TO 4
      N=IX(J)+JX
      A(N)=SYM(2)
      YLAB=Y(J)
      IFLG=1
    4 CONTINUE
      IF(IFLG.EQ.0) GO TO 11
      WRITE(6,FRMT)(A(L),L=1,NSKP),YLAB,(A(L),L=JX,110)
      GO TO 12
   11 WRITE(6,5) (A(L),L=1,110)
   12 CONTINUE
    3 CONTINUE
      Do 8 I=1,JX
    8 A(I)=SYM(1)
      DO 9 I=JX,110
    9 A(I)=SYM(4)
      WRITE(6,5) (A(I),I=1,110)
      CALL TOP(NDIV,XMIN,XMAX,A,NFL,NDEC,JX,IXPTS)
C
C     BOTTOM LABEL IS SETHERE.
C
      NCHR=22
      ENCODE(22,205,TLAB)
  205 FORMAT(22HSURFACE ELEVATIONS CM.)
      CALL TLB(TLAB,NCHR,IXPTS,JX)
      RETURN
      END


      SUBROUTINE SLB(SB,IS,IE,IYPTS)
      DIMENSICN SE?(1)
      NCHF=27
      SB(1)=1HD
      SB(2)=1HI
      SB(3)=1HS
      SB(4)=1HT
      SB(5)=1H.
      SB(6)=1H
      SB(7)=1HA
      SB(8)=1HL
      SB(9)=1HO
      SB(10)=1HN
      SB(11)=1HG
      SB(12)=1H
      SB(13)=1HB
      SB(14)=1HO
      SB(15)=1HU
      SB(16)=1HN
      SB(17)=1HD
      SB(18)=1HA
      SB(19)=1HR
      SB(20)=1HY
      SB(21)=1H
      SB(22)=1H1
      SB(23)=1H0
      SB(24)=1H
      SB(25)=1HK
      SB(26)=1HM
      SB(27)=1H.
      IH=IYPTS/2
      JH=NCHR/2
      IS=IH-JH
      IE=IS+NCHR-1
      RETURN
      ENO
```

```
      FUNCTION JFIX(R)
      JFIX=IFIX(R)
      D=R-FLOAT(JFIX)
      IF(D.GE..5) JFIX=JFIX+1
      RETURN
      END
```

```
      SUBROUTINE ISBNDRY(I,NO)
      COMMON/CUTOFF/IBP,NFLX,JJ1,JJ2
      COMMON/BOUND/IB(75),BV(75),NBV
C
C        SUBROUTINE TO TELL US IF WE ARE AT A BOUNDARY POINT, ANO
C        WHAT TYPE OF BOUNDARY POINT.
C
C     NO=0, NOT A BOUNDARY POINT.
C     NO=-1,DIRICHLET BOUNDARY POINT,ON OPEN BOUNDARY.
C     NO=1,ONSHORE BOUNDARY POINT INCLUDING ISLAND.
C
      NIBP=IBP
      IF(NIBP.LT.0) NIBP=-IBP
      IST=NFLX+NIBP+1
      IEND=NIBP+NFLX
      IS=NIBP+1
      IF(IEND.EQ.0) GO TO 4
      IF(NIBP.EQ.0) GO TO 2
      DO 1 J=1,NIBP
      K=IB(J)
      IF(I.EQ.K) GO TO 8
    1 CONTINUE
    2 IF(NFLX.EQ.0) GO TO 4
      DO 3 J=IS,IEND
      K=IB(J)
      IF(I.EQ.K) GO TO 7
    3 CONTINUE
    4  DO 5 J=IST,NBV
      K=IB(J)
      IF(I.EQ.K) GO TO 6
    5 CONTINUE
      NO=0
      RETURN
    6 NO=-1
      RETURN
    7 NO=1
      RETURN
    8 NO=1
      RETURN
      END
```

```
      SUBROUTINE TLB(TLAB,NCHR,IXPTS,JX)
      DIMENSION FRMT(3),TLAB(3)
      JHALF=NCHR/2
      IHALF=IXPTS/2
      NSKP=JX+IHALF-JHALF
      NC=10
      ENCODE(16,100,FRMT)NSKP,NC
      WRITE(6,FRMT)TLAB(1),TLAB(2),TLAB(3)
  100 FORMAT(5H(1H,,I2,4HX,3A,I2,1H))
      RETURN
      END
```

```
      SUBROUTINE SCS(I1,I2,S,CS,DS)
      COMMON/IX/X(200)
      COMMON/IY/Y(200)
      DX=X(I2)-X(I1)
      DY=Y(I2)-Y(I1)
      DS=(DX*DX+DY*DY)**.5
      S=DY/DS
      CS=DX/DS
      RETURN
      END
```

```
      SUBROUTINE MATRIX(DSHAPEX,DSHAPEY,K,NVRTX,NTRI, IFILE,MASL)
C     SUBROUTINE TO FORM GLOBAL MATRIX AND RIGHT HAND SIDE
      DIMENSION DSHAPEX(1),DSHAPEY(1)
      COMMON/IHEIGHT/N(200)
      COMMON/ICONST/CONST1,CONST2
      COMMON/CUTOFF/NISP,NFLX,NO,JNO
      COMMON/IGRAD/DALPHAX,DALPHAY,DDEPTHX,DDEPTHY,DDELTAX,DDELTAY,AREA
      COMMON/IWORK/VALP(4500)
      COMMON/INT/INTP(4500)
      COMMON/IRHS/RHS 1200)
      COMMON/IIP/IP(3,350)
      COMMON/WIND/TAUX,TAUY,CURL
      NROW=NVRTX
      DO 1 I=1,3
      II=IP(I,K)
C
C
      IF(N(II).EQ.-1) GO TO 1
      IF(N(II).EQ.1) GO TC 1
      IF(NO.EQ.-1) GO TO 4
      IF(IFILE)3,3,4
    3 DO 2 J=1,3
      JJ=IP(J,K)
      CALL MNSRT(II,JJ,INTP,NB,NROW,MASL)
      VALP(NB)=VALP(NB)+(+CONST2*(DSHAPEX(I)*DSHAPEX(J)+DSHAPEY(I)*DSHAP
     *EY(J))+1./3.*(DSHAPEY(J)*DDEPTHX-DSHAPEX(J)*DDEPTHY))*AREA
    2 CONTINUE
    4 CONTINUE
      CHK=CONST1*(DALPHAY*DDEPTHX-DALPHAX*DDEPTHY) /3,
      CHK1=CONST1*CONST2*(DSHAPEX(I)*DALPHAX+DSHAPEY(I)*DALPHAY)
      RHS(II)=RHS(II)+(-CHK-CHK1-1./3.*CURL)● AREA
    1 CONTINUE
  900 FORMAT(*0*,*TRI.*,I3,5X,*VERTX.*,I5 *5X9*JACOBO**F1OO4)
      RETURN
      ENO
```

```
  103 CONTINUE
  102 CONTINUE
      WRITE(6,200) I1,I2
  200 FORMAT(*0*,*WE ARE LOST ALONG POINTS*,2I5,2X,*CAN NOT FIND BOUNDAR
     CY TRIANGLE IN MATRIX ASSEMBLY ROUTINE*)
      GO TO 101
C
C     WE HAVE FOUND TRIANGLE, NOW TO GET GRADIENTS
C
  104 DO 106 L=1,3
      M=IP(L,K)
      A(L,1)=X(M)
      A(L,2)=Y(M)
      A(L,3)=1.
  106 CONTINUE
      CALL ALPHX(K,DALPHX,DALPHY,ALPHA,SA,ALPH)
      CALL ALPHX(K,DDELTX,DDELTY,DELTA,SD,ALPH)
      CALL SCS(I1,I2,S,CS,DS)
      HMEAN=-(DEPTH(I1)+DEPTH(I2))/2.
      DDELTS=CS*DDELTX+S*DDELTY
      DALPHN=-S*DALPHX+CS*DALPHY
      DALPHS=CS*DALPHX+S*DALPHY
      TS=CS*TAUX+TAUY*S
C
      DO 107 L=1,3
      DO 108 M=1,3
  108 B(M)=0.
      B(L)=1.
      CALL GRAD(B(1),B(2),B(3),DSHPX(L),DSHPY(L),CSHPE(L))
  107 CONTINUE
C
C     NOW ADD TO GLOBAL MATRIX
C
      II=I1
  109 IF(NEWBV.GT.1) GO TO 111
      IF(IFILE)609,609,111
  609 DO 112 L=1,3
      JJ=IP(L,K)
      CALL MNSRT(II,JJ,INTP,NB,NVRTX,MASL)
      DSHPN=-S*DSHPX(L)+CS*DSHPY(L)
      DSHPS=CS*DSHPX(L)+S*DSHPY(L)
      VALP(NB)=VALP(NB)-(CONST2*(DSHPN+DSHPS)-HMEAN*DSHPS) ● OS/20
  112 CONTINUE
  111 CONTINUE
C
C     NOW ADD CONTRIBUTION TO RHS
C
      RHS(II)=RHS(II)-(CONST1*DDELTS-CONST1*CONST2*(DALPHN+DALPHS)-TS)*D
     CS/2.
      IF(II.EQ.I2) GO TO 101
      II=I2
      GO TO 109
  101 CONTINUE
```

```
      SUBROUTINE BC(NVRTX,NBV,IB,BV,MASL)
C
u     SUBROUTINE TO ADO ON BOUNDARY VALUE CONTRIBUTION
C
C     THE BOUNDARYROWS HAVE ALREADY BEENZEROEDDURING THE ASSEMBLY
C     PROCESS*  THEREFCRE HE PROCEED TO FILL THE RIGHT HAND SIDE
C     VECTOR WITH THE BOUNDARY CONDITIONS AND SET THEDIAGNOL
c     ELEMENTS TO ONE IF IT OCCURS IN A BOUNDARY ROW.
C
      DIMENSION IB(1),BV(1)
      COMMON/CUTOFF/NIBP,NFLX,JJ1,JJ2
      COMMON/INT/INTP(4500)
      COMMON/IWORK/VALP(4500)
      COMMON/IRHS/RHS(200)
      NROW=NVRTX
      NSTP=NFLX+NIBP+1
      Do 10 K=NSTP,NBV
      J=IB(K)
      RHS(J)=BV(K)
      CALL MNSRT(J,J,INTP,NB,NROW,MASL)
      VALP(NB)=1.
   10 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SETB(RHS,VALP)
      DIMENSION RHS(1),VALP(1)
      COMMON/INT/INTP(4500)
      COMMON/CUTCFF/NIBP,NFLX,JJ1, JJ2
      COMMON/BOUND/IB(75),BV(75),NBV
      COMMON/NUMB/NVRTX,NTRI,IFILE,MASL,MEWBV
      IF(NIBP.EQ.0) GO TO 5
      KL=IB(1)
      RHS(KL)=BV(1)
      DO 1I=1,NVRTX
      CALL MNSRT(KL,I,INTP,NB,NVRTX,MASL)
      VALP(NB)=0.
      IF(I.EQ.KL) VALP(NB)=1.
    1 CONTINUE
    5 ISTRT=NIBP+1
      I1=IB(ISTRT)
      RHS(I1)=BV(ISTRT)
      00 7 I=1,NVRTX
      CALL MNSRT(I1,I,INTP,NB,NVRTX,MASL)
      VALP(NB)=0.
      IF(I1oEQoI) VALP(NB)=1.
    7 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE BASS(VALP,RHS)
      DIMENSION VALP(1),RHS(1)
      DIMENSION DSHPX(3),DSHPY(3),CSHPE(3)
      COMMON/ICCNST/CONST1,CONST2
      COMMON/INT/INTP (45001
      COMMON/LSCOEF/SA(4),SD(4)
      COMMON/IDELTA/DELTA(200)
      COMMON/IALPHA/ALPHA(200)
      COMMON/IDEPTH/DEPTH(200)
      COMMON/IY/Y(200)
      COMMON/IIP/IP(3,350)
      COMMON/IA/A(3,3)
      COMMON/IB/B(3)
      COMMON/IX/X(200)
      COMMON/NUMB/NVRTX,NTRI,IFILE,MASL,NEWBV
      COMMON/BOUND/IB(75),BV(75),NBV
      COMMON/IALP/ALPH(3)
      COMMON/WIND/TAUX,TAUY,CURL
      COMMON/CUTOFF/NIBP,NFLX,JJ1,JJ2
C
C     NOW IMPOSE THE NO FLUX BOUNDARY CONDITIONS,
C
C     BEGIN WITH ISLAND.
C
      IF(NIBP.EQ.0.AND.NFLX.EQ. O) RETURN
      IF(NIBP.EQ.0) GO TO 1
      ISTART=1
      ISTOP=NIBP-1
      GO TO 2
    1 ISTART=1
      ISTOP=NFLX-1
     'GO TO 2
    3 ISTART=NIBP+1
      ISTOP=NIBP+NFLX-1
    2 CONTINUE
C
C     NOW LOCATE BOUNDARY SIDE.
C
      DO 101 I=ISTART,ISTOP
      J=I+1
      I1=IB(I)
      I2=IB(J)
c
G     NOW LOCATE THE TRIANGLE WE ARE IN
c
      DO 102 K=1,NTRI
      KF=0
      JF=0
      DO 103 L=1,3
      II=IP(L,K)
      IF(II.EQ.I1) JF=L
      IF(II.EQ.I2) KF=L
      IF(KF.NE.0.AND.JF.NE.0) GO TO 104
```

```
      SUBROUTINE SLVK(NROW,NCOL,ERROR,INTP,VALP)
C.........SOLUTIONOFLINEAR SYSTEM AFTER QLUP DECOMPOSITION
C.........KNIUTHORTHOGONAL LIST STRAGE SCHEME
C.........
C.........NROW IS NUMBER OF ROWS IN MATRIX
C.........NCOL IS NUMBER OF COLUMNS IN MATRIX
C.........ERROR IS AN INTEGRR VARIABLE RETURNED AS 1
C.........IF AN ERROR OCCURS(ATTEMPTED DIVISION BY ZERO)
C.........OTHERWISE 2
C.........INTP IS AN INTEGR ARRAY OF LENGTH MSZ*((44/(NUMBER OF BITS
C.........PER INTEGER WORD)+1)...(SEE HINIT)
C.........VALF IS A REAL ARRAY OF LENGTH MSZ
C.........THE RIGHT HANO SIDE MUST BE STORED IN (VALP(I),I=1,2,....,NROW)
C.........AND THE SOLUTION IS RETURNED IN (VALP(I),I=NROW+1,NROW+2,...
C.........,NROW+NCOL)
C.........
C.........CAN ACCEPT A MATRIX WITH BOTH PERMUTED ROWS ANO PERMUTED COLS
C.........**31APR75
      INTEGER INTP(1),ERROR
      REAL VALP(1)
      IP=IUP(INTP,1)
      IVS1=LEFT(INTP,NROW+1)+NROW
      VALP(IVS1)=VALP(IP)
      I=1
   80 IF(I-NCOL)10,20,20
   10 I=I+1
      IP=IUP(INTP,I)
      IM1=I-1
      SUM=0.0
C.........ROW SCAN FROM COL1 TO COL IM1
      ILFT=IP
   50 ILFT=LEFT(INTP,ILFT)
      KCOLS=ICOL(INTP,ILFT)
      IF(KCOLS) 60,60,30
   30 KCOLA=ICOL(INTP,KCOLS+NROW)
      IF(KCOLA-IM1) 70,70,50
   70 KCOLV=KCOLS+NROW
      SUM=SUM+VALP(ILFT)*VALP(KCOLV)
      GOTO 50
   60 IV=LEFT(INTP,I+NROW)+NROW
      VALP(IV)=VALP(IP)-SUM
      GOTO 80
   20 IP=IUP(INTP,NCOL)
      NB=NBOX(IP,LEFT(INTP,NCOL+NROW),NROW,INTP)
      IF(NB) 100,100,110
  100 ERROR=1
      RETURN
  110 NV=LEFT(INTF,NROW+NCOL)+NROW
      VALP(NV)=VALP(NV)/VALF(NB)
C.........COLUMN SCAN FROM ROW NCOL-1 TO ROW i
      I=NCOL
  170 IF(I-1) 150,150,160
  160 I=I-1


      IF(NIBP.EQ.0) GO TO 201
      J=NIBP-1
      IF(ISTOP.EQ.J) GO TO 3
  zoi CALL SETB(RHS,VALP)
      RETURN
      END
```

```
      SUBROUTINE DCPK(NROW,NCOL,SING,RANK,FILL, SMALL,INTP,VALP,MASL)
C..........QLUP DECOMPOSITION OF A REAL MATRIX
C..........KNUTH ORTHOGONAL LIST STORAGE
C..........
C..........NROW IS NUMBER CF ROWS IN MATRIX
C..........NCOL IS NUMBER OF COLUMNS IN MATRIX
C..........ROW SCALE FACTORS ARE STORED IN ROW BASE VALUES TEMPORARILY
C..........SING IS AN INTEGER VARIABLE RETURNING 1 IF MATRIX IS SINGULAR
C..........MEANING THAT RANK IS LESS THAN NCOL ...SING RETURNS 2 OWISE
C..........RANK IS AN INTEGER VARIABLE RETURNING THE ROW RANK OF THE MATR
C..........FILL IS AN INTEGER VARIABLE RETURNING THE NUMBER OF ORIGINALLY
C..........ZERO MATRIX ELEMENTS THAT BECAME NONZERO CURING DECOMPOSITION
C..........MINUS THE NUMBER CF INITIALLY NONZERO ELEMENTS THAT BECAME
C..........ZERO DURING THE DECOMPOSITION PROCESS
C..........SMALL IS A REAL CCNSTATNT THAT SPECIFIES THE SMALLEST ABSOLUTE
C..........VALUE OF AN ELEMENT RELATIVE TO THE LARGEST ABSOLUTE VALUE
C..........CF ANY ELEMENT IN ITS ROW BEFORE DECOMPOSITION (THE ROW NORM)
C..........THAT WILL BE REPRESENTED BY MATRIX STORAGE ELEMENT
C..........INTP IS AN INTEGER ARRAY OF LENGTH MSZ*(44/(INTEGER WORD LENGT
C..........IN BITS)+1....(SEE MINIT SUBROUTINE)
C..........MASL IS A VARIABLE INITIALIzE BY MINIT
C..........ROWS ANO COLUMNS ARE PERMUTED VIA ROW+COL BASES
C***** #***
C..........**31MAR76
C..........
      REAL VALP(1)
      INTEGER INTP(1), SING,RANK,FILL
      FILL=0
C..........FIND SCALE FOR EACH ROW =1.0/(MAX ABS VALUE IN ROW)
      I=0
   90 IF(I-NROW) 10,20,20
   10 I=I+1
      ROWNR=0.0
C..........START ROW I SCAN
      ILFT=I
   50 ILFT=LEFT(INTP,ILFT)
      IF( ICCL(INTP,ILFT)) 30,30,40
   40 ABV=ABS(VALP(ILFT))
      IF(ABV-ROWNR) 50,50,60
   60 ROWNR=ABV
      GOTC 50
   30 IF(ROWNR-SMALL) 70,70,80
   70 VALP(I)=0.0
      GOTC 90
   80 VALP(I)=1.0/ROWNR
      GOTO 90
   20 NM1=NCOL-1
C..........START COLUMN SCAN
      K=0
  240 IF(K-NM1) 100,110,110
  100 K=K+1
C..........SCAN LOWER RIGHT SUBMATRIX FOR COLUMN WITH LEAST NUMBER OF
C..........NONZERO ELEMENTS BYT AT LEAST ONE
```

```
      SUBROUTINE PACK(I,J,VALP,VAL,INTP,NROW,MASL )
      DIMENSICN VALP(1),INTP(1)
      CALL MNSRT(I,J,INTP,NB,NROW,MASL)
      IF(MASL.EQ.-1) GC TC 10
      VALP(NB)=VAL
      RETURN
   10 WRITE(6,5)I,J
    5 FORMAT(*0*,*OVERFLOW CF VALP AT ENTRY*,I5,*,*,I3)
      RETURN
      END
```

```
      GOTO 650
C........TOOSMALL SO DELETE
  690 CALL MDLET(IP,ICJRS,NROW,INTP,MASL)
      FILL=FILL-1
      GOTO 650
C.........NO MATCH SO FILL IN ONE
  660 VAL=EM*VALP(JP)
C.........UNLESS TOOSMALL TO STORE
      IF(ABS(VAL)*SKALE-SMALL) 650,650,710
  710 FILL=FILL+1
      CALL MNSRT(IP,ICJFS,INTP,NB,NROW,MASL)
C.........DEYECTMEMORU OVERFLOW
      IF(NB) 720,720,730
  730 VALF(NB)=VAL
      GOTO 650
C.........MEMORY OVERFLOW
C.........USER DETECTS THIS CONDITION IN CALLING PRIGRAM
C.........BY TESTING WHETHER MASL.LE. O
  720 K=1
      GOTO 180
  110 IF(NBOX(IUP(INTP,NCCL),LEFT(INTP,NCOL+NROW),NROW,INTP))500,500,510
C.........RANK IS NCOL-1 IF DIAGONAL IS ZERO
  500 K=NCCL
      GOTC 180
  510 RANK=NCOL
      SING=2
      RETURN
      END
```

```
      IP=IUP(INTP,I)
      IP1=I+1
      SUM=0.0
C.........ROW SACN FROM CCL IP1 TO COL NCOL
      ILFT=IP
  200 ILFT=LEFT(INTP,ILFT)
      KCOLS=ICOL(INTP,ILFT)
      IF(KCOLS) 180,180,190
  190 KCOLA=ICOL(INTP,KCOLS+NROW)
      IF(KCOLA-IP1) 200,310s320
  320 IF(KCOLA-NCOL) 31O*3IO,2OO
  310 KCOLV=KCOLS+NROW
      SUM=SUM+VALF(ILFT)*VALP(KCOLV)
      GOTO ZOO
  180 NB=NBOX(IP,LEFT(INTP,I+NROW),NROW,INTP)
      IF(NB) 100,100,220
  220 IV=LEFT(INTP,I+NROW)+NROW
      VALP(IV)=(VALF(IV)-SUM)/VALP(NB)
      GOTO 170
  150 ERRCR=2
      RETURN
      END
```

```
      SUBROUTINE SUP(INTP,NBOX,IUP)
      INTEGER INTP(1)
      CALL PAC(INTP(NBOX),20,35,IUP)
      RETURN
      END
```

```
      NEMAX=NCOL+1
      NPIV=0
      KA=K-1
  620 IF(KA-NCOL)800,810,810
  800 KA=KA+1
      CALL FVSL1(KA,NROW,NCOL,INTP,VALP,IPIV,NE)
      IF(NE) 820,820,830
  830 IF(NE-NEMAX) 840,820,820
  840 NPIV=IPIV
      NEMAX=NE
      GOTO 820
  810 IF(NPIV)180,180,190
C.........NO PIVOT CN K-TH COLUMN
  180 RANK=K-1
      SING=1
      RETURN
C.........PIVOT ON IPIV
  190 KP=IROW(INTP,NPIV)
      KC=ICOL(INTP,NPIV)
C.........ROW INTERCHANGE IF PIVOT NOT IN ROW K
      CALL MRXA(K,IROW(INTP,KP),NROW,INTP)
C.........COLUMN INTERCHANGE IF PIVOT NOT IN COLUMNK
      CALL MCXA(K,ICOL(INTP,KC+NROW),NROW,INTP)
      PIVOT=VALP(NPIV)
      I=K
  230 IF(I-NROW)220,240,240
  220 I=I+1
      IP=IRP(INTP,I)
      NB=NEOX(IP,LEFT(INTP,K+NROW),NROW,INTP)
      IF(NB) 230,230,250
  250 EM=-VALF(NB)/PIVOT
      VALF(NB)=-EM
C.........START SCAN OF PIVOT ROW IN STORED OROER
      JP=KP
      SKALE=VALP(IP)
  650 JP=LEFT(INTP,JP)
      ICJPS=ICOL(INTP,JP)
      IF(ICJPS) 23U,23O,E1O
  610 ICJPA=ICOL(INTP,ICJPS+NROW)
      IF(ICJPA-K) 650,550,630
  630 IF(ICJPA-NCOL)640,640,650
C.........TRY TO FIND ACTUAL SAMECOLUMN IN WORKROW
  640 JR=IP
  600 JR=LEFT(INTP,JR)
      ICJRS=ICOL(INTP,JR)
      IF(ICJRS) 660,660,670
  670 ICJRA=ICOL(INTP,ICJRS+NROW)
      IF(ICJRA-ICJPA) 600,680,600
C.........FIND MATCH
  680 VAL=VALP(JR) +EM*VALP(JP)
      IF(ABS(VAL)*SKALE-SMALL ) 690,690,700
C.........STORE RESULT
  700 VALF(JR)=VAL
```

```
      Subroutine MNSRT(KROW,KCOL,INTP,NB,NROW,MASL)
C.........INSERTS A MATRIX STORAGE ELEMENT AT ROW KROW AND COLUMN KCOL
C.........UNLESS THERE IS ALREADY AN ELEMENT ALLOCATED IN WHICH CASE
C.........THAT ELEMENT NUMBER IS RETURNED IN NB
C.........INTP IS AN INTEGER ARRAY
C..............(SEEE MINIT)
C.........NB IS THE NEW ELEMENT NUMBER
C.........NROW IS THE NUMBER OF ROWS IN THE MATRIX
C.........MASL IS AN INTEGER VARIABLE INITIALIZED BY MINIT
C..............MASL=-1 IMPLIES OVERFLOW IN WHICH CASE NB
C..............RETURNS -1
C.Q..* **.***.*.
      INTEGER INTP(1)
      IRGT=KROW
   30 ILFT=LEFT(INTP,IRGT)
      IF (ICOL(INTP,ILFT)-KCOL) 50,20,90
   90 IRGT=ILFT
      GOTO 30
   50 IBLW=NROW+KCOL
  150 IABV=IUP(INTP,IBLW)
      IF(IROW (INTP,IABV)-KROW) 130,20,120
  120 IBLW=IABV
      GOTO 150
  130 IF(MASL) 60,60,70
   60 NB=-1
      RETURN
   70 NB=MASL
      MASL=LEFT(INTP,MASL)
      CALL SLEFT(INTP,IRGT,NB)
      CALL SLEFT(INTP,NB,ILFT)
      CALL SUP(INTP,IBLW,NB)
      CALL SUP(INTP,NB,IABV)
      CALL SROW(INTP,NB,KROW)
      CALL SCOL(INTP,NB,KCOL)
      RETURN
   20 NB=ILFT
      RETURN
      END
```

```
      SUBROUTINE SCOL(INTP,NBOX,ICOL)
      INTEGER INTP(1)
      CALL PAC(INTP(NBOX),48,59,ICOL)
      RETURN
      END
```

```
      SUBROUTINE SROW(INTP,NBOX,IROW)
      INTEGER INTP(1)
      CALL PAC(INTP(NBOX),36,47,IROW)
      RETURN
      END
```

```
      FUNCTION LEFT(INTP,NBOX)
      INTEGER INTP(1)
      CALL UNPAC(INTP(NBOX),4,19, LEFT)
      RETURN
      END
```

```
      FUNCTION IUP(INTP,NBOX)
      INTEGER INTP(1)
      CALL UNPAC(INTP(NBOX),20,35,IUP)
      RETURN
      ENO
```

```
      FUNCTION ICOL(INTP,NBOX)
      INTEGER INTP(1)
      CALL UNPAC(INTP(NBOX),48,59,ICOL)
      RETURN
      END
```

```
      FUNCTION IRCH(INTP,NBOX)
      INTEGER INTP(1)
      CALL UNPAC(INTP(NBOX),36,47,IROW)
      RETURN
      ENO
```

```
      SUBROUTINE MDLET(KROW,KCOL,NROW,INTP,MASL)
C.........DELETES MATRIX STORAGE ELEMENT AT IROW,ICOL IF ONE  EXISTS
C.........NROW IS THE NUMBER OF ROWS IN THE MATRIX
C.........INTP IS AN INTEGER ARRAY (SEE MINIT)
C.........MASL IS AN INTEGER VARIABLE INITAILIZED BY MINIT
C.........******020176******
C..............
      INTEGER INTP(1)
      IABV0=NROW+KCOL
      ILFT0=KROW
   30 ILFT=LEFT(INTP,ILFT0)
      IC=ICOL(INTP,ILFT)
      IF(IC) 10,10s20
   20 IF(IC-KCOL) 70,40,70
   70. ILFT0=ILFT
      GOTO 30
   40 IABV=IUP(INTP,IABV0)
      IR=IROW(INTP,IABV)
      IF(IR) 10,10,50
   50 IF(IR-KROW) 60,60s50
   80 IABV0=IABV
      GOTO 40
   60 ILFT1=LEFT(INTP,ILFT)
      CALL SLEFT(INTP,ILFT0,ILFT1)
      IABV1=IUP(INTP,IABV)
      CALL SUP(INTP,IABV0,IABV1)
      CALL SLEFT(INTP,IABV,MASL)
      MASL=IABV
   10 RETURN
      END
```


```
      FUNCTION IRCH(INTP,NBOX)
      INTEGER INTP(1)
      CALL UNPAC(INTP(NBOX),36,47,IROW)
      RETURN
      ENO
```

```
      FUNCTION NBOX(KROW,KCOL,NROW,INTP)
C.........GETS MATRIX S1ORAGE ELEMENT NUMBER IF ONE EXISTS FOR ROW KROW
C.........AND COLUMN KCOL
C.........NROW IS HE NUMBER OF ROWS IN THE MATRIX
C.........INTP IS AN INTEGER ARRAY (SEE MINIT)
C.........NBOX RETURNS THE ELEMENT NUMBER UNLESS ONE DOESNT EXIST
C.........IN WHICH CASE IT RETURNS -1
C..............
      INTEGER INTP(1)
      ILFT=KROW
   30 ILFT=LEFT(INTP,ILFT)
      IF(ICOL (INTP,ILFT)-KCOL) 10,20,30
   10 NBOX=-1
      RETURN
   20 NBOX=ILFT
      RETURN
      END
```

```
      SUBROUTINE SLEFT(INTP,NBOX,LEFT)
      INTEGER INTP(1)
      CALL FAC(INTP(NBOX),4,19,LEFT)
      RETURN
      END
```

```
      SUBROUTINE MRXA(KRA,JRA,NROW,INTP)
      INTEGER INTP(1)
      IF(KRA-JRA) 10,20,10
   20 RETURN
   10 KRS=IUP(INTP,KRA)
      JRS=IUP(INTP,JRA)
      CALL SUP(INTP,KRA,JRS)
      CALL SUP(INTP,JRA,KRS)
      CALL SROW(INTP,JRS,KRA)
      CALL SROW(INTP,KRS,JRA)
      RETURN
      ENO
```

```
      SUBROUTINE LQFIT(VAL,S,AVAR,LIST,NPTS)
      COMMON/IRHS/B(200)
      COMMON/IDEPTH/D(200)
      COMMON/IHEIGHT/STNDV(200)
      DIMENSION A(4,4)
      DIMENSION VAL(1),S(1),C(4),IPS(1)
C
C     LEAST SQUARES FIT TO THIRD ORDER POLYNOMIAL IN Z.
C
C     S ARE THE COEFFICIENTS,LIST IS THE LISTINGS OPTION.
C
      NCOEF=4
      DO 2 I=1,NCOEF
      B(I)=0.
      DO 3 J=1,NCOEF
    3 A(I,J)=0.
    2 CONTINUE
      DO 4 I=1,NPTS
      C(1)=D(I)*D(I)*D(I)
      C(2)=D(I)*D(I)
      C(3)=D(I)
      C(4)=1.
      DO 11 J=1,NCOEF
      DO 7 K=J,NCOEF
    7 A(J,K)=A(J,K)+C(J)*C(K)
   11 B(J)=B(J) +VAL(I)*C(J)
    4 CONTINUE
C
C     NOW TO FILL THE OTHER HALF OF THE MATRIX
      DO 8 I=1,NCOEF
      DO 9 J=1,NCOEF
    9 A(J,I)=A(I,J)
    8 CONTINUE
      CALL INVR(A,4,B,1,DETERM,4,4)
      DO 306 I=1,4
      S(I)=B(I)
  306 CONTINUE
      IF(LIST.EQ.0) GO TO 101
      WRITE(6,20) (S(I),I=1,NCOEF)
   20 FORMAT(*1*,*COEFFICIENTS ARE*,2X,5F10.4)
      IF(LIST.LT.0) GO TO 201
      WRITE(6,45)
   45 FORMAT(*0*,*GLOBAL LABEL*,10X,*DEPTH*,10X,*ALPHA*,10X,*DEVIATION*,
     *10X,*VARIANCE*,/)
      GO TO 144
  201 WRITE(6,145)
  145 FORMAT(*0*,*GLOBAL LABEL*,10X,*DEPTH*,10X,*DELTA*,10X,*DEVIATION*,10
     *10X,*VARIANCE*,/)
  144 CONTINUE
  101 TOT=0.
      DO 13 I=1,NPTS
      ALP=S(1)*D(I)*D(I)*D(I)+S(2)*D(I)*D(I)+S(3)*D(I)+S(4)
      SD=VAL(I)-ALP

      STNDV(I)=SD
      VAR=SD*SD
      TOT=TOT+VAR
      IF(LIST.EQ.0) GO TO 102
      WRITE(6,35)I,D(I),VAL(I),SD,VAR
   35 FORMAT(* *,I7,13X,F8.2,7X,F10.4,6X,F9.3,10X,F9.3)
  102 CONTINUE
   13 CONTINUE
      AVAR=TOT/FLOAT(NPTS)
      AVAR=AVAR**.5
      IF(LIST.EQ.0) GO TO 103
      WRITE(6,30) AVAR
   30 FORMAT(*0*,*MEAN STANDARD DEVIATION MAGNITUDE IS*,2X,F8.3)
  103 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE MCXA(KCA,JCA,NROW,INTP)
      INTEGER INTP(1)
      IF(KCA-JCA) 10,20,10
   20 RETURN
   10 KCS=LEFT(INTP,KCA+NROW)
      JCS=LEFT(INTP,JCA+NROW)
      CALL SLEFT(INTP,JCA+NROW,KCS)
      CALL SLEFT(INTP,KCA+NROW,JCS)
      CALL SCCL(INTP,KCS+NRCH,JCA)
      CALL SCCL(INTP,JCS+NRCH,KCA)
      RETURN
      ENO
```

```
      SUBROUTINE SLCPE(X1,Y1,X2,Y2,SLOP,DIST)
      DY=Y2-Y1
      DX=X2-X1
      IF(DX.EQ.0)DX=.00000001
      SLOP=DY/DX
      DIST=((DX*DX)+(DY*DY))**.5
      RETURN
      END
```

```
      SUBROUTINE FINDSID(VAL1,VAL2,VAL3,VAL,IP1,IP2,IFLG)
      DIMENSICN IP1(2),IP2(2)
      IFLG=0
      J=1
      IF(VAL1OEO,VALI GO TO 11
      IF(VAL2.EQ.VAL) GO TO 12
      IF(VAL3.EQ.VAL)GO TO 1 3
    6 IF(VAL.LT.VAL1.AND.VAL.GT.VAL2) GO TO 1
      IF(VAL.GT.VAL1.AND.VAL.LT.VAL2) GO TO 1
    7 IF(VAL.LT.VAL2.AND.VAL.GT.VAL3) GO TO 2
      IF(VAL.GT.VAL2.AND.VAL.LT.VAL3) GO TO 2
    8 IF(VAL.LT.VAL1.AND.VAL.GT.VAL3) GO TO 3
      IF(VAL.GT.VAL1.AND.VAL.LT.VAL3) GO TO 3
      IF(IFLG.EQ.1) GO TO 4
      WRITE(6,5)
    5 FORMAT(*1*,*WE HAVE SCREWED UP IN FINDING THE TRIANGLE SIDE*)
      GO TO 4
   11 IP1(1)=1
      IP2(1)=2
      J=2
      IFLG=1
      GO TC 7
   12 IP1(1)=1
      IP2(1)=2
      J=2
      IFLG=1
      GO TO 8
   13 IP1(1)=2
      IP2(1)=3
      J=Z
      IFLG=1
      GO TO 6
    1 CONTINUE
      IP1(J)=1
      IP2(J)=2
      IF(J.GT.1) GO TO 4
      J=J+1
      IFLG=0
      GO TC 7
    2 IP1(J)=2
      IP2(J)=3
      IF(J.GT.1) GO TO 4
```

```
      J=J+1
      IFLG=0
      GO TO 8
    3 IP1(J)=1
      IP2(J)=3
      IFLG=0
    4 RETURN
      END


      SUBROUTINE SWITCH(A,B)
      C=A
      A=B
      B=c
      RETURN
      END


      SUBROUTINE FINDPT(VAL1,VAL2,VAL,X1,Y1,X2,Y2,X,Y,SLPE)
      ISWITCH=0
      IF(X2.GT.X1) GO TO1
      ISWITCH=1
      CALL SWITCH(VAL1,VAL2)
      CALL SWITCH(X1,X2)
      CALL SWITCH(Y1,Y2)
    1 CONTINUE
      DIST12=VAL1-VAL2
      DIST=VAL1-VAL
      RATIO=DIST/DIST12
      DIST12=X1-X2
      DIST=Y1-Y2
      DIST12=(DIST12*DIST12+DIST*DIST)**.5
      DIST=RATIO*DIST12
      IF(X1.EQ.X2) GO TO 11
      ANGLE=ATAN(SLPE)
      DX=DIST*COS(ANGLE)
      DY=DX*SLPE
      GO TO 12
   11 CONTINUE
      DX=0.
      IF(Y1.GT.Y2)DIST=-DIST
      DY=DIST
   12 CONTINUE
      X=X1+DX
      Y=Y1+DY
      IF(ISWITCH.EQ.0) GO TO 2
      CALL SWITCH(VAL1,VAL2)
      CALL SWITCH(X1,X2)
      CALL SWITCH(Y1,Y2)
    2 RETURN
      ENO


      SUBROUTINE CHECK(VAL1,VAL2,VAL3,VAL,ICHK,IP1,IP2)
      ICHK=1
      IF(VAL1.EQ.VAL2.AND.VAL1.EQ. VAL) GO TO 1
      IF(VAL2.EQ.VAL3.AND.VAL2.EQ. VAL) GO TO 2
      IF(VAL1.EQ.VAL3.AND.VAL1.EQ.VAL) GO TO 3
      ICHK=0
      IP1=0
      IP2=0
      GO TO 4
    1 IP1=1
      IP2=2
      GO TO 4
    2 IP1=2
      IP2=3
      GO TO 4
    3 IP1=1
      IP2=3
    4 RETURN
      END
```

```
      SUBROUTINE KONTRI(X,Y,IP,CON,NTRI,NVRTX,NCON,VALUE,NPEN)
      DIMENSION X(1),Y(1),IP(3,1),CON(1),VALUE(1),X1(2),Y1(2),III(2),JJJ
     *(2),AX(2),AY(2)
C
C     THIS IS A LINEAR CONTOURING ROUTINE USING THE TRIANGLES
C     X AND Y ARE COORDINATES OF THE TRIANGLE VERTICES
C     IP IS THE MATRIX(3 X NTRI) CONTAINING THE GLOBAL LABELS OF EACH
C     TRIANGLE VERTEX
C     CON IS THE VECTOR CONTAINING THE CONTOUR INTERVALS.
C     NTRI IS THE NUMBER OF TRIANGLES
C     NVRTX IS THE NUMBER OF GLOBAL POINTS
C     NCON IS THE NUMBER OF CONTOURS YOU HAVE.  IF THIS IS LESS THAN
C     ZERO, THEN YOU ARE READING IN THE CONTOUR INTERVALS WITH
C     THE NUMBER OF INTERVALS EQUAL TO ABS(NCON).  IF NCON IS
C     POSITIVE, THEN THE PROGRAM WILL GENERATE THE CONTOUR INTERVALS
C     BY DIVIDING THE RANGE OF VALUES EVENLY INTO NCON INTERVALS.
C     VALUE IS THE VECTOR CONTAINING THE VALUES TO BE CONTOURED AT EACH
C     VERTEX
C     NPEN IS THE PEN NUMBER YOU WANT TO USE FOR CONTOURING
      CALL STPEN(NPEN)
      IF(NCON.LT.0) GO TO 3
      VMAX=0.
      VMIN=1000000.
      DO 1 I=1,NVRTX
      VMAX=AMAX1(VMAX,VALUE(I))
    1 VMIN=AMIN1(VMIN,VALUE(I))
      CALL CONINT(VMAX,VMIN,NCON,CON)
    3 NCON=IABS(NCON)
      N=1
   11 I=IP(1,N)
      J=IP(2,N)
      K=IP(3,N)
      ALARG=AMAX1(VALUE(I),VALUE(J),VALUE(K))
      ASMAL=AMIN1(VALUE(I),VALUE(J),VALUE(K))
      IFIR=1
      DO 6 L=1,NCON
      IF(CON(L).LT.ASMAL.OR.CON(L).GT.ALARG) GO TO 6
      IF(IFIR.GT.1) GO TO 4
      CALL SLOPE(X(I),Y(I),X(J),Y(J),S12,DIS12)
      CALL SLOPE(X(J),Y(J),X(K),Y(K),S23,DIS23)
      CALL SLOPE(X(I),Y(I),X(K),Y(K),S13,DIS13)
      IFIR=2
    4 CALL CHECK(VALUE(I),VALUE(J),VALUE(K),CON(L),ICHK,II,JJ)
      IF(ICHK.EQ.0)GO TO 5
      II=IP(II,N)
      JJ=IP(JJ,N)
      X1(1)=X(II)
      Y1(1)=Y(II)
      X1(2)=X(JJ)
      Y1(2)=Y(JJ)
      CALL STNPTS(2)
      CALL SLLILI(X1,Y1)
      GO TO 6


      SUBROUTINE CONINT(VMAX,VMIN,INT,C)
      DIMENSION C(1)
      INT=INT+1
      DIFF=VMAX-VMIN
      AINT=DIFF/FLOAT(INT)
      INT=INT-1
      DO 1 I=1,INT
      C(I)=VMIN+FLOAT(I)*AINT
    1 CONTINUE
      WRITE(6,2)
    2 FORMAT('1', 'CONTOUR INTERVALS ARE')
      DO 4 I=1,INT
    4 WRITE(6,3)I,C(I)
    3 FORMAT('0',I5,3X,F10.2)
      RETURN
      END
```

```
      SUBROUTINE SETUP(ISTART,IPRINT,XSIZE,YSIZE,TLABEL, IAXIS,XMIN, XMAX,
     *YMIN,YMAX,NDIVX,NDIVY,XSTRT,YSTRT)
      DIMENSICN TLABEL(3)
C     THIS IS A GENERAL SUBROUTINE TO SETUP AN NPS PROGRAM
C     ISTART IS ZERO IF THIS IS THE FIRST TIME YOU CALL THE NPS ROUTINES
C     IF YCU HAVE ALREADY CALLEO PRNTON OR STCCON PREVIOUSLY,ISTART=1
C     IPRINT=0 WILL CALL PRINTER PLOT
C     IPRINT=1 WILL CALL STCCON
C     SXIZE IS THE LENGHT OF PLOT IN INCHES IN X-DIRECTION
C     YSIZE IS LENGHT OF PLOT IN INCHES IN Y-DIRECTION
C     TLABEL IS ENCCDED IN MAIN PROGRAH WITH 30 SPACES AND DIMENSION 3.
C     IT WILL BE THE TOP LABEL OF PLOT,
C     IF YCU WANT AXIS DRAWN UP AND LABELED, IAXIS IS 1, OTHERWISE IT
C     IS ZERO.
C     XMIN,YMIN,XMAX,YMAX, ARE THE MAXIHUH AND MINIMUM X AND Y VALUES
C     TO BE USEO TO LABEL THE AXIS ANO SET UP THE SUBJECT SPACE.
C     NDIVX ANO NDIVY ARE THE NUMBER OF DIVISIONS YOU WANT THE AXIS
C     LABELING TO SHOW
C
      IF(ISTART.EQ.1) GO TO 2
      IF(IPRINT.EQ.1) GO TO 1
      CALL PRNTON
      GO TO 2
    1 CALL STCCON(48HCALCOMP PLOT OF WHATEVER NEEDS TO BE PLOTTED
    2 CONTINUE
      CALL STPEN(1)
      XADD=XSTRT+XSIZE
      YADD=YSTRT+YSIZE
      CALL STS20B(XSTRT,XADD,YSTRT ,YADD)
      CALL STSUBJ(XMIN,XMAX,YMIN,YMAX)
      CALL STNDIV(1,1)
      CALL GDLILI
      HEIGHT=XSIZE/30.
      IF(HEIGHT.GT..49) HEIGHT=.49
      CALL STCHSZ(HEIGHT)
      CALL STNCHR(30)
      CALL STLNOR(0.)
      CALL TITLET(TLABEL)
      CALL AXLILI
      IF(IAXIS.EQ.0) GO TO 3
      CALL STNDIV(NDIVX,NDIVY)
      HEIGHT=HEIGHT/2.
      CALL STCHSZ(HEIGHT)
      CALL STNDEC(1)
      CALL NODLIB
      CALL NODLIL
    3 RETURN
      ENO
```

```
5 CALL FINDSID(VALUE(I),VALUE(J),VALUE(K),CON(L),III,JJJ,IFLG)
  IF(IFLG.EQ.1) GO TO 6
  IA=III(1)
  II=III(2)
  JA=JJJ(1)
  JJ=JJJ(2)
  IF (IA.EQ.1.AND.JA.EQ.2) SLP1=S12
  IF(IA.EQ.2. AND.JA.EQ. 1) SLP1=S12
  IF(II.EQ.1.AND.JJ.EQ.2) SLP2=S12
  IF(JJ.EQ.1.AND.II.EQ.2) SLP2=S12
  IF(IA.EQ.1.AND.JA.EQ.3) SLP1=S13
  IF(JA.EQ.1.AND.IA.EQ.3) SLP1=S13
  IF (II.EQ.1.AND.JJ.EQ.3) SLP2=S13
  IF (JJ.EQ.1.AND.II.EQ.3) SLP2=S13
  IF(IA.EQ.2. AND.JA.EQ.3) SLP1=S23
  IF(JA.EQ.2.AND.IA.EQ.3) SLP1=S23
  IF(II. EQ.2.AND.JJ.EQ.3) SLP2=S23
  IF(JJ.EQ.2. AND.II.EQ.3) SLP2=S23
  IA=IP(IA,N)
  II=IP(II,N)
  JA=IP(JA,N)
  JJ=IP(JJ,N)
  CALL FINDPT(VALUE(IA),VALUE(JA),CON(L),X(IA),Y(IA),X(JA),Y(JA),X1 (
 *1),Y1(1),SLP1)
  CALL FINDPT(VALUE(II),VALUE(JJ),CON(L),X(II),Y(II),X(JJ),Y(JJ),X1(
 *2),Y1(2),SLP2)
  CALL STNPTS(2)
  CALL SLLILI(X1,Y1)
6 CONTINUE
  N=N+1
  IF(N.GT.NTRI) GO TO 7
  GO TO 11
7 RETURN
  END



  SUBROUTINE VRTXLB(X,Y,VAL,NDEC,NVRTX,HEIGHT,NPEN)
  DIMENSION X(1),Y(1),VAL(1)
C     THIS SUBROUTINE LABELS THE POINTS
C     THE COORDIANTES OF THE POINTS ARE GIVEN BY X AND Y
C     THE VALUE AT EACH POINT IS GIVEN BY VAL
C
  CALL STPEN(NPEN)
  CALL STCHSZ(HEIGHT)
  CALL STLNOR(0)
  CALL STNDEC(NDEC)
  DO 1 I=1,NVRTX
  CALL STLNST(X(I),Y(I))
  CALL DECVAL(VAL(I))
1 CONTINUE
  RETURN
  END
```

```
      SUBROUTINE DRTRI(X,Y,IP,NTRI,NPEN,N)
      DIMENSION X(1),Y(1),IP(3,1),N(1),ZXX(2),ZYY(2)
C
C     SUBROUTINE TO DRAW SUEROUTINES
C     X IS X-ARRAY
C     Y IS Y-ARRAY
C     IP IS A THREE BY NTRI MATRIX GIVING THE GLOBAL LABELS OF EACH VRTX
C     N IS A STORAGE ARRAY MAKING SURE WE DONT'T DRAW A LINE TWICE
C     N SHOULD BE LARGER THAN THE AMOUNT OF LINES YOU HAVE
C     NPEN IS THE PEN NUMBER YOU WANT TO USE
C
      CALL STPEN(NPEN)
      N(1)=0
      M=1
      LL=O
      DO 1 I=1,NTRI
      IX=1
      IY=2
    2 II=IP(IX,I)
      IJ=IP(IY,I)
      ZXX(1)=X(II)
      ZYY(1)=Y(II)
      ZXX(2)=X(IJ)
      ZYY(2)=Y(IJ)
      ICHECK=II*II*II+IJ*IJ*IJ+II*II+IJ*IJ+II+IJ
      DO 4 L=1,M
      IF(ICHECK.EQ.N(L))GO TO 6
    4 CONTINUE
      M=M+1
      IF(M.LT.301) GO TO 10
      M=3C0
      LL=LL+1
      IF(LL.GT.300) LL=1
      N(LL)=ICHECK
      Go 70 11
   10 CONTINUE
      N(M)=ICHECK
   11 CONTINUE
      CALL STNPTS(2)
C                                                    )4(RTXTTS LLAC
      CALL STTXTR(1)
      CALL SLLILI(ZXX,ZYY)
    6 IF(IY.EQ.3)GO TO B
      IY=3
      GO TO 2
    B IF(IX.EQ.2)GO TO 1
      IX=2
      GO TO 2
    1 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE DYEBLAN(BRT,BRC,CURL,BFRIC,CONST1)
      COMMON/IGRAD/DALPHAX,DALPHAY,DDEPTHX,DDEPTHY,DEX,DEY,AREA
      BRT=-(DEY*DDEPTHX-DEX*DDEPTHY)
      BRC=-(DALPHAY*DDEPTHX-DALPHAX*DDEPTHY)*CONST1
      BFRIC=-BRT-BRC-CURL
      RETURN
      END
```

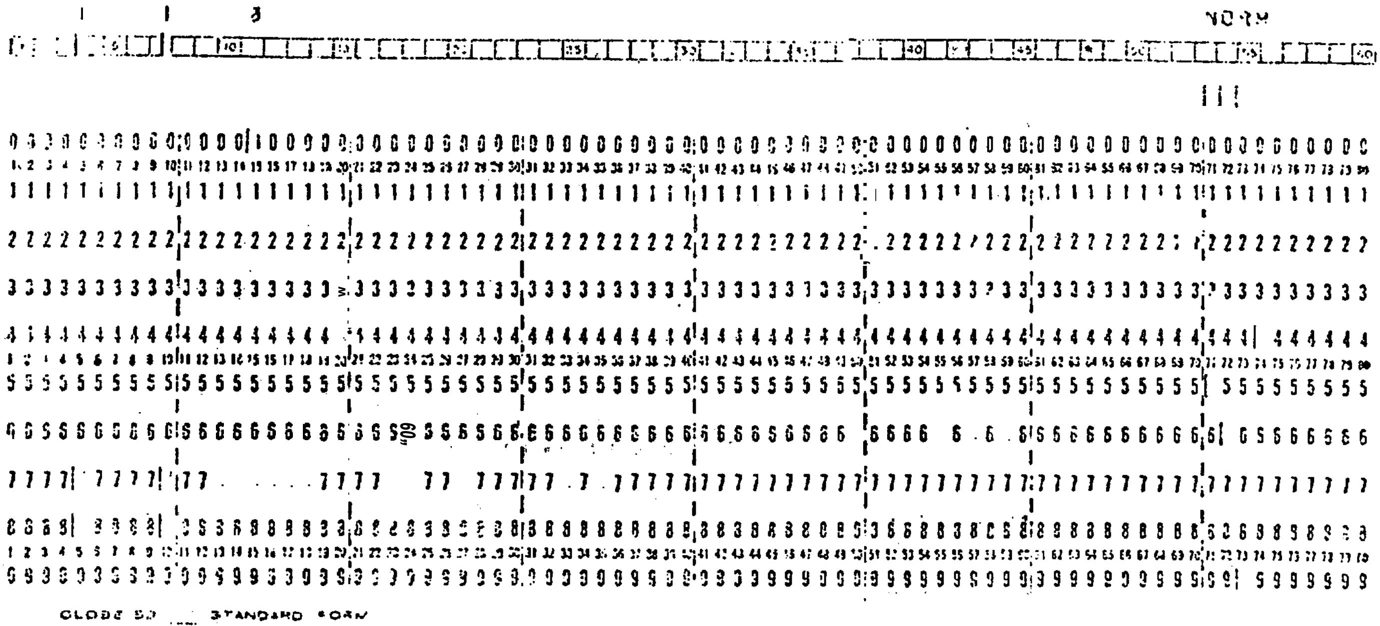
```
      SUBROUTINE TRILABL(X,Y,IP,NTRI,HEIGHT,NPEN)
      DIMENSION X(1),Y(1),IP(3,1)
C
C     THIS SUBROUTINE LABELS THE TRIANGLES
C     X AND Y ARE RESPECTIVE COORDINATES
C     IP IS THE 3 X NTRI MATRIX CONTAINING THE GLOBAL LABELS OF THE VRTX
C     NTRI IS THE NUMBER OF TRIANGLES
C     HEIGHT IS THE HEIGHT OF THE NUMBERS IN INCHES
C     NPEN IS THE PEN NUMBER
C
      CALL STPEN(NPEN)
      CALL STCHSZ(HEIGHT)
      DO 1 I=1,NTRI
      II=IP(1,I)
      IJ=IP(2,I)
      IK=IP(3,I)
      EX=(X(II)+X(IJ)+X(IK))/3.
      EY=(Y(II)+Y(IJ)+Y(IK))/3.
      XSHIFT=(HEIGHT/4.)*1.25
      YSHIFT=(HEIGHT/7.)*1.5
      EX=EX-XSHIFT
      EY=EY-YSHIFT
      CALL STNDEC(0)
      CALL STLNST(EX,EY)
      CALL DECVAL(FLOAT(I))
    1 CONTINUE
      RETURN
      END


      SUBROUTINE BAROT(U,V,TOT)
      COMMON/IGRAD/DALPHAX,DALPHAY,DDEPTHX,DDEPTHY,DEX,DEY,AREA
      COMMON/SCALES/USCALE,DSCALE,ALSCALE,G,E,Q,GAMMA,FO,EDDY
      U=-DEY*USCALE
      V=DEX*USCALE
      U=U*100.
      V=V*100.
      TOT=(U*U+V*V)**.5
      RETURN
      END


      SUBROUTINE EKMAN(U,V,TOT)
      COMMON/WIND/TX,TY,CURL
      COMMON/SCALES/USCALE,HSCALE,ALSCALE,G,E,Q,GAMMA,FO,EDDY
      C=(EDDY*Q*FO)**(-.5)
      U=C*(TX+TY)*USCALE*FO*HSCALE*Q
      V=C*(- TX+TY)*USCALE*FO*HSCALE*Q
      U=U*10000.
      V=V*10000.
      TOT=(U*U+V*V)**.5
      RETURN
      END


      SUBROUTINE BOTT(U,V,TOT)
      COMMON/IGRAD/DALPHAX,DALPHAY,DDEPTHX,DDEPTHY,DEX,DEY,C
      COMMON/SCALES/USCALE,DSCALE,ALSCALE,G,E,Q,GAMMA,FO,EDDY
      C=G/(Fe+@)
      U=-C*(DALPHAY*E*DSCALE+DEY*FO*USCALE*Q*ALSCALE/G)/ALSCALE
      V=C*(DALPHAX*E*DSCALE+DEX*FO*USCALE*ALSCALE*Q/G)/ALSCALE
      U=U*100.
      V=V*100.
      TOT=(U*U+V*V)**.5
      RETURN
      END
```

```
      SUBROUTINE SURF(UE,VE,UB,VB,U,V,TOT)
      U=UE+UB
      V=VE+VB
      TOT=(U*U+V*V)**.5
      RETURN
      END
```

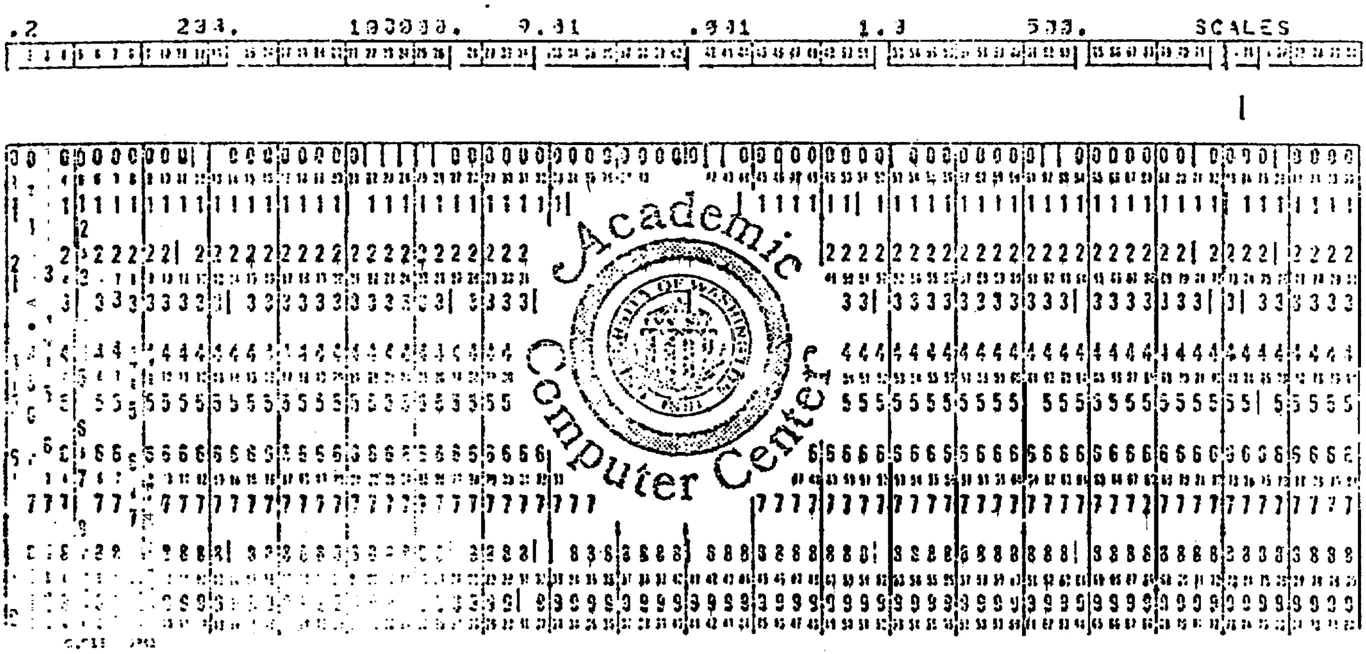
```
* BITS I J OF A ARE EXTRACTED
*IF I GT 5, N=60-(I-J-1)
*BITS I - 59, 0 - J OF A ARE EXTRACTED
*BIT J OF A ALWAYS GOES TO BIT 59 OF B
*                                    J. J. THOMAS  4/7/75
          ENTRY  UNPAC
          VFD    42/0LUNPAC
          IFEQ   *F,2
          VFD    18/UNPAC     FTN ARGUMENT LINKAGE
UNPAC     BSSZ   1
          SB7    1            B7 = 1
          SA2    A1+B7        GET ADDRESSES OF I - B IN X: - X
          SA3    A2+B7
          SA4    A3+B7
R         MICRO  1,1,$X$
          ELSE
          IFNE   *F,1,1
          ERR                 NOT CALLED BY RUN OR FTN
          VFD    18/4         RUN 2,3 ARGUMENT LINKAGE
UNPAC     BSSZ   1
R         MICRO  1,1,$B$
          ENDIF
          SA2    "R".2        X2 = I
          SA3    "R".3        X3 = J
          IX5    X3-X2        FIGURE NO. BITS - ... EXTRACT FROM A
          PL     X5,J1
          SX5    X5+60
J1        SB6    X2-60
          SB5    X3+1
          SB7    X5
          MX0    1            MAKE MASK OF CORRECT LENGTH
          SA1    "R".1        X1 = A
          AX0    B7
          LX0    -B6          ALIGN MASK WITH CORRECT BITS OF A
          BX6    X0*X1        EXTRACT BITS FROM A
          LX6    B5           RIGHT-JUSTIFY THEM
          SA6    "R".4        STORE IN B
          EQ     UNPAC        EXIT
          END
```

```
          ICENT  PAC            CALL  PAC(1,I,J,B)
● PACS THE N RIGHTMOST BITS OF 9 INTO A
•BITS ARE NUMBERED LEFT TO RIGHT FROM 0 - 59
* IF I LE J, N=J-I+1
•DESTINATION IS BITS I - J OF A
•IF I GT J, N=60-(I-J-1)
* DESTINATION IS BITS I - 59, 0 - J OF A
'BIT 59 CF B ALWAYS GOES TO 5?7 J OF A
•                               J. J. THOMAS 4/7/75
          ENTRY  PAC
          VFD    42/0LPAC
          IFEQ   ● F,2
          VFD    18/PAC         FTN ARGUMENT LINKAGE
PAC       BSSZ   1
          SB7    1              B7 = 1 (CONSTANT)
          SA2    A1+B7          GET ADDRESSES OF A - B IN X1 - X4
          SA3    A2+B7
          SA4    A3+B7
R         MICRO  1,1,$X$
          ELSE
          IFNE   ● F?IP1
          ERR                   NOT CALLED BY RUN OR FTN
          VFD    18/4           RUN 2,3 ARGUMENT LINKAGE
PAC       BSSZ   1
          SB7    1              B7 = 1 (CONSTANT)
R         MICRO  1,1,$B$
          ENDIF
          SA1    vRv.1          X1 = A
          SA2    vRv.2          X2 = I
          SA3    vRv.3          X3 = J
          SA4    vRv.4          X4 = B
          1X5    X3-X2          FIGURE NO. BITS - 1 TO PAC INTO A
          MX0    1
          PL     X5,J1
          SX5    X5+60
J1        SB6    x5
          AX0    06             MAKE MASK OF CORRECT LENGTH IN RIGHT-MOST
          SB7    B6+B7          PART "OF WORD
          LXO    $37
          BX4    X0*X4          EXTRACT BITS FROM B
          SB7    x3-59
          LX0    -87            ALIGN BITS FROM B AND MASK
          LX4    -B7            WITH CORRECT BITS OF A
          BX1    -X0*X1         INSERT BITS FROM B INTO A
          8X6    X1+X4
          SA6    A1             STORE A
          EQ     PAC            EXIT
          END
          IDENT  UNPAC          CALL UNPAC(A,I,J,B)
" EXTRACTS N BITS A ANO PLACES THEM
' IN B , RIGHT-JUSTIFIED WITH ZERO FILL
* BITS ARE NUMBERED LEFT TO RIGHT FROM O - 59
* IF I LE J, N=J-I+1
```

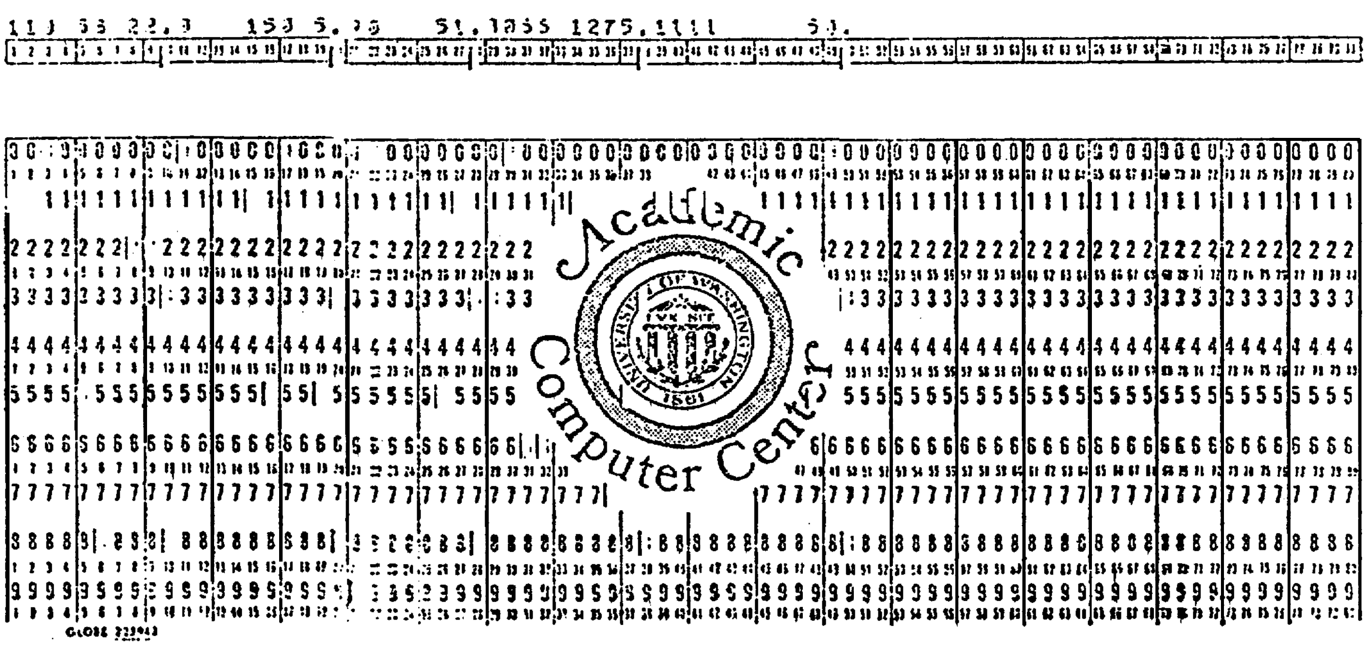
APPENDIX III SAMPLE CARDS



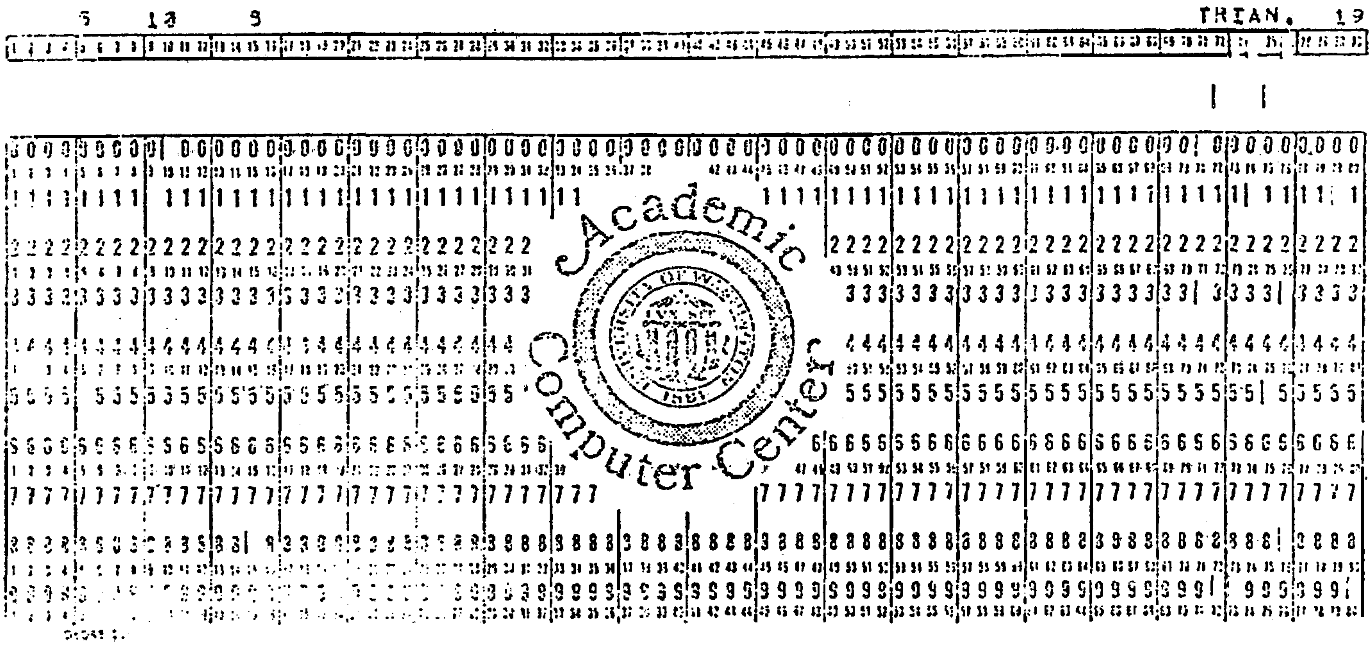
Control Card



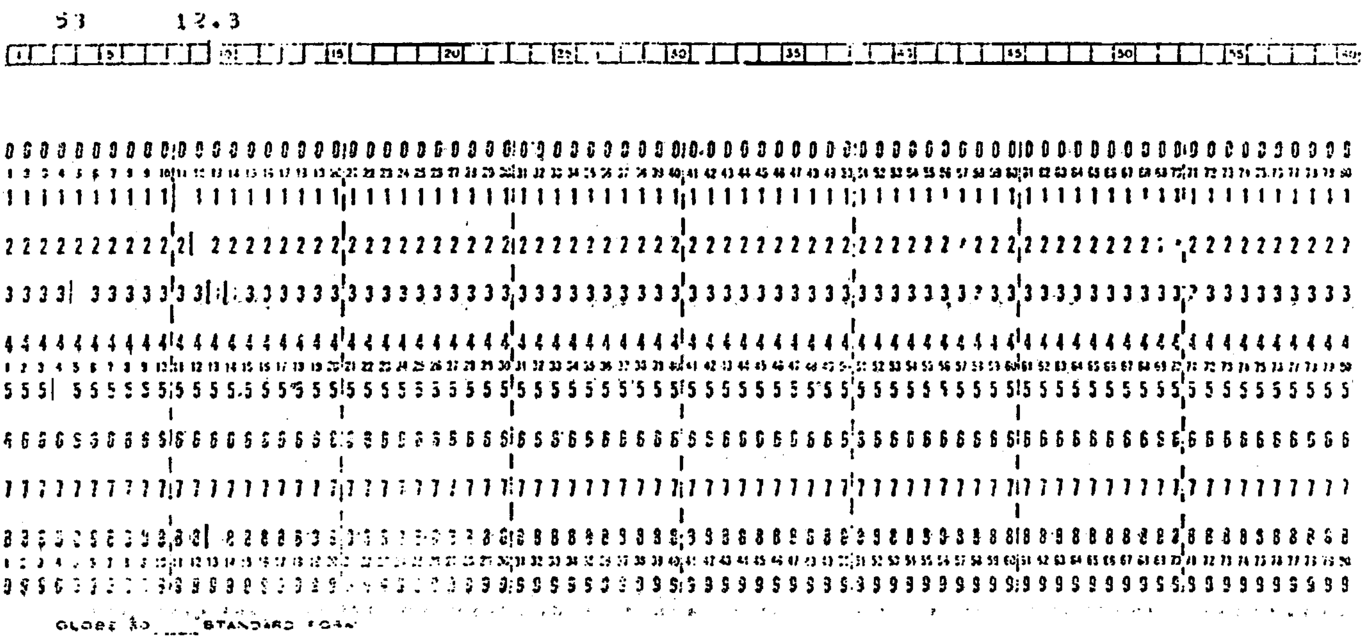
Scale Parameters



Raw Station Data



Triangle

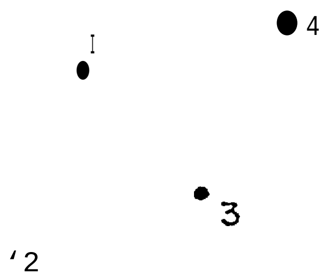


Boundary Condition Card

APPENDIX IV

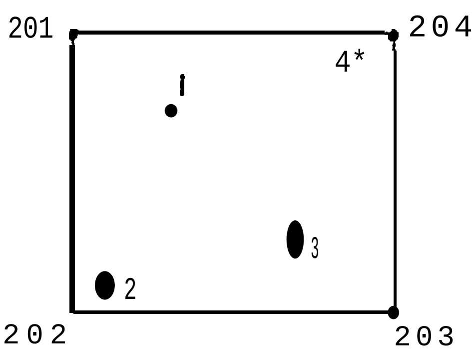
TRIANGLE SCHEME

The triangle routine used was originally devised by Smyth (1975) and later simplified by Galt. To demonstrate how it works, we use a simple 4 station example, shown below':



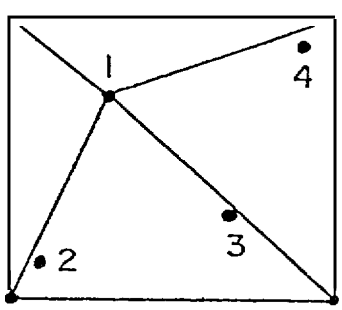
The 4 stations to be triangulated.

The first step is to enclose the region of interest by a rectangle:



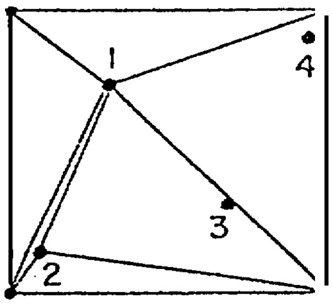
Rectangle defined by artificial points 201, 202, 203, and 204 enclose region of interest.

Next, take the first point and use that to subdivide the rectangle:



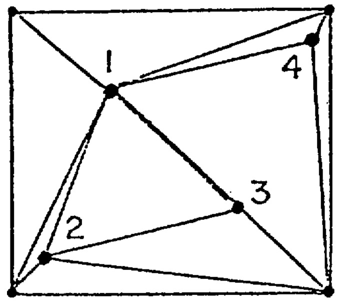
First subdivision of the initial triangle.

The second point is then used to subdivide the triangle it is in:



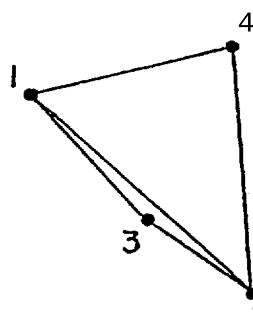
Second point used to subdivide the triangle.

After all of the points have been used to subdivide the larger triangles they lie in, we have:

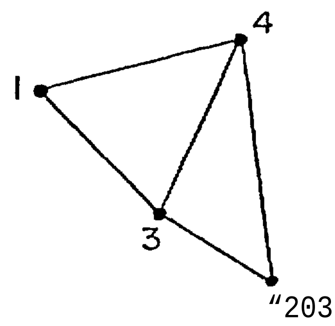


Mesh after all the subdivisions have been made.

Next, several sweeps are made to check the "goodness" of pairs of triangles. For example, the pair of triangles:

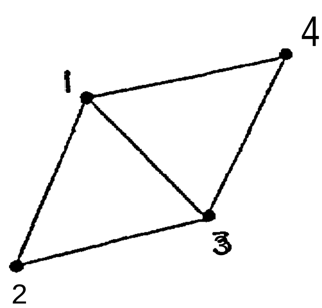


is checked to see if the triangles would be more equilateral if subdivided in the following way:



From tests run on a set of 130 stations and about 200 triangles, it was found that after about five sweeps through the entire mesh, the method converged and yielded a “best” mesh.

The last step in generating the mesh is to eliminate all triangles with corner vertices (201, 202, 203, 204). The final mesh is shown below:



Final mesh.

The routine has been written to accommodate up to 200 stations including one set of interior boundary points (an island). If there are interior boundary points, they must be read in first and in clockwise order.

To evaluate

$$N_2 \iint_D \phi_j \nabla^2 \phi_i \, dxdy, \qquad (A1)$$

Begin by evaluating

$$N_2 \iint_D \phi_j \frac{\partial^2 \phi_i}{\partial x^2} \, dxdy.$$

First integrate

$$\int \phi_j \frac{\partial^2 \phi i}{\partial x^2} \, dx \qquad (AZ)$$

by parts to get

$$\int \phi_j \frac{\partial^2 \phi_i}{\partial x^2} \, dxdy = \phi_j \frac{\partial \phi_i}{\partial x} - \int \frac{\partial \phi_j}{\partial x} \frac{\partial \phi_i}{\partial x} \, dx \qquad (A3)$$

Now integrate by $\int dy$ to get

$$N_2 \iint_D \phi_j \frac{\partial^2 \phi_i}{\partial x^2} \, dxdy = N_2 \left\{ \int \phi_j \frac{\partial \phi_i}{\partial x} \, dy - \iint \frac{\partial \phi_i}{\partial x} \frac{\partial \phi_j}{\partial x} \, dxdy \right\} \quad (A4)$$

The second term of (A1) can be integrated by parts also, to give

$$N_2 \iint \phi_j \frac{\partial^2 \phi_i}{\partial y^2} \, dxdy = N_2 \left\{ \int \phi_j \frac{\partial \phi_i}{\partial y} \, dx - \iint_D \frac{\partial \phi_i}{\partial y} \frac{\partial \phi_j}{\partial y} \, dydx \right\}. \quad (A5)$$

Putting (A4) and (A5) together gives

$$N_2 \iint \phi_j \nabla^2 \phi_i \, dxdy = N_2 \left\{ \int \phi_j \frac{\partial \phi_i}{\partial x} \, dy + \int \phi_j \frac{\partial \phi_i}{\partial y} \, dx - \iint_D \frac{\partial \phi_i}{\partial x} \frac{\partial \phi_j}{\partial x} \quad \frac{\partial \phi_j}{\partial y} \frac{\partial \phi_i}{\partial y} \, dxdy \right\} \qquad (A6)$$

Now use the following substitution into the first two terms on the right-hand side:

$$dy = \frac{\partial y}{\partial s} \, ds \;, \; dx = \frac{\partial x}{\partial s} \, ds, \qquad (A7)$$

To integrate $\int_{\Delta} \phi\, dxdy$

$$\phi = Ax + By + C \text{ such that}$$

$$\phi = 1 \text{ at } (X\ ,\ Y\ ) \text{ and}$$

$$\phi = 0 \text{ at } (0,0) \text{ and } (b,0)$$

Begin by integrating Part I first:

$$\phi = \left(\frac{y}{y_2}\right)$$

$$\delta x = \frac{x_2-b}{y_2}\, y + (b-x_2)$$

$$\int_{\Delta I} \phi\, dxdy = \int_0^{y_2} \left(\frac{y}{y_2}\right) \left(\frac{x_2-b}{y_2}\, y + [b-x_2]\right) dy$$

$$= (b-x_2) y_2 \frac{1}{6}.$$

Now to integrate Part II:

$$\phi = \frac{y}{y_2}$$

$$\delta x = \frac{x_2}{y_2}\, y + x_2$$

$$\int_{\Delta\, II} \phi\, dxdy = \int_0^{y_2} \left(\frac{y}{y_2}\right) \left(- \frac{x_2}{y_2}\, y + x_2\right) dy$$

$$= x_2 y_2 \quad \frac{1}{6}$$

The combined results give

$$\int_A \phi\, dxdy = \frac{1}{6}\, y_2 (b-x_2+x_2) = \frac{1}{6}\, y_2 b$$

$$= \frac{1}{3} \text{ (area of triangle).}$$