A GENERAL METHOD FOR GENERATING BATHYMETRIC DATA FOR HYDRODYNAMIC COMPUTER MODELS

By Jon R. Burau and Ralph T. Cheng

U.S. GEOLOGICAL SURVEY

Open-File Report 89-28

Prepared in cooperation with the

CALIFORNIA STATE WATER RESOURCES CONTROL BOARD and the

CALIFORNIA DEPARTMENT OF WATER RESOURCES

8035-54

Sacramento, California
1989

DEPARTMENT OF THE INTERIOR

MANUEL LUJAN, JR., Secretary

U.S. GEOLOGICAL SURVEY

Dallas L. Peck, Director

For additional information
write to:

District Chief
U.S. Geological Survey
Federal Building, Room W-2234
2800 Cottage Way
Sacramento, CA 95825

Copies of this report may be
purchased from:

U.S. Geological Survey
Books and Open-File Reports Section
Box 25425
Building 810, Federal Center
Denver, CO 80225

CONTENTS

ILLUSTRATIONS

COVER: A computer-generated graph illustrates the shape of the bottom of San Pablo Bay. Vertical scale is exaggerated a factor of 800.

CONVERSION FACTORS

Metric (International System) units are used in this report. For those readers who prefer to use the inch-pound system, the conversion factors for the terms used in this report are listed below:

Multiply metric unit	By	To obtain inch-pound unit
meter (m)	3.281	feet (ft)
kilometer (km)	0.6214	mile (mi)

A GENERAL METHOD FOR GENERATING BATHYMETRIC DATA FOR HYDRODYNAMIC COMPUTER MODELS

By Jon R. Burau and Ralph T. Cheng

ABSTRACT

This report describes a general method for generating water-depth data from randomly distributed bathymetric data for numerical hydrodynamic models. Raw input data from field surveys, water-depth data digitized from nautical charts, or a combination of the two are sorted to give an ordered data set on which a search algorithm is used to isolate data for interpolation. Water depths at locations required by hydrodynamic models are interpolated from the bathymetric data base using linear or cubic shape functions used in the finite-element method. The bathymetric data-base organization and preprocessing, the search algorithm used in finding the bounding points for interpolation, the mathematics of the interpolation formulae, and the features of the automatic generation of water depths at hydrodynamic model grid points are discussed. This report includes documentation of two computer programs which are used to (1) organize the input bathymetric data and (2) to interpolate depths for hydrodynamic models. An example of computer program operation is drawn from a realistic application to the San Francisco Bay estuarine system.

INTRODUCTION

Realistic hydrodynamic models of bays and estuaries require accurate representations of the bathymetry of the embayment. In general, preparation of input bathymetric data for modeling is tedious and time consuming. The algorithms and associated computer programs described in this report provide a streamlined method for constructing detailed bathymetric data for input to hydrodynamic models. Raw input data from field surveys, water-depth data digitized from nautical charts, or a combination of the two are sorted to give an ordered data set on which a search algorithm is used to isolate data for interpolation. Water depths at locations required by hydrodynamic models are interpolated from the bathymetric data base using linear or cubic shape functions from the finite-element method. The bathymetric data-base organization and preprocessing, the search algorithm used in finding the bounding points for interpolation, the mathematics of the interpolation formulae, and the features of the automatic generation of water depths at hydrodynamic model grid points are discussed. This report prepared by the U.S. Geological Survey in cooperation with the California State Water Resources Control Board and the California Department of Water Resources includes documentation of two computer programs which are used to (1) organize the input bathymetric data and (2) to interpolate depths for hydrodynamic models. In this report, the preprocessing part of this method is discussed first, followed by a description of how bathymetric grids are generated. Finally, the interpolation algorithms are discussed along with a description of the computer code and its usage.

Most commonly used interpolation algorithms (Barnhill, 1977; Franke and Neilson, 1980) applied in surface approximation invoke statistically motivated averaging techniques which have little physical justification. In some statistical algorithms, a large number of data points often are used to ensure that a given interpolation is "well represented" but not necessarily bounded by the data. Surface approximations at points that are "well represented" but are not bounded by the data are essentially extrapolated values with a higher probability of incorrect representation. Even if the interpolation point is bounded using these techniques, the effect of the actual bounding points may be minimal because of the averaging used. Because the computational effort associated with defining the bounding relations is complex and extensive, most existing algorithms do not explicitly ensure their interpolations are

based on data that locally bound the location in question. Indeed, isolating the bounding points on which the approximations are based from the overall data base is the most CPU intensive task in this entire method.

Practical application of the method for generating water-depth data is a three-step process. The first step involves collecting bathymetric data at known locations. Data entered into this system consist of a series of spatial coordinates, x,y,z, that define the surface of the bottom of an embayment, where x,y,z can be referenced to any convenient orthogonal right-handed coordinate system. In this report, x and y define the horizontal plane, and the z coordinate defines the depth of the embayment. Unlike many methods that require the spatial location of the known data to fall on a regular orthogonal grid, depth data can be entered into this system in a completely random fashion. This feature provides simplified data entry that can accommodate bathymetric data collected from a variety of sources using differing techniques. For example, this system can easily incorporate data generated directly from boat surveys, or, when direct bathymetric data are not available, data collected from nautical charts. A large quantity of data can be quickly generated for modeling studies by digitizing the bathymetric contours on nautical charts. The randomly distributed depth information typically supplied on the charts can be used to fill in the gaps between contours on the charts.

The second step in the application of the method for generating water-depth data involves the creation of a data base by sorting the data and by making preliminary calculations. This is done by a computer program, which also is used to edit the data base. After the data base is created, the third and final step involves a separate computer program that interpolates depths for hydrodynamic models that use either finite-difference grids or finite-element grids. Additionally, cross-sectional information of a basin can be generated using this program.

Interpolations in this method are based on a local surface that bounds the interpolation point x^*,y^* by a triangle of known data points. The interpolated depth, z^*, is defined locally within the triangle of known data by either linear or cubic shape functions used in the application of the finite-element method (Lapidus and Pinder, 1982).

BATHYMETRIC DATA BASE - PREPROCESSING

In order to increase the efficiency of the interpolations, certain calculations are performed by a preprocessing computer program and stored along with the basic topological x,y,z data. The data are sorted first according to the y-coordinate magnitude using a simple "paired interchange sort" (Cole, 1978); subsequent to that, point densities (to be defined) are calculated. The program then deletes any data points that have the same x,y coordinates, keeping only the first occurrence at a given location. Finally, surface gradients at each data point are estimated using linear triangles.

Calculation of the Point Densities

The local point density is denoted as $\frac{\partial J}{\partial y}(J)$, where J is the counter for the sorted data $J=1,2,..J_{max}$. The point densities are used in the search algorithm (to be discussed in "Location of Bounding Points" section) and represent the rate of change of the data base pointer, ΔJ, by the corresponding distance in the sort, or y direction, Δy. If a change in the data base pointer, ΔJ, is represented by N in the sorted data base, the following relation is used to calculate the point densities:

$$\frac{\partial J}{\partial y}(J) = \frac{\Delta J}{\Delta y}(J) = \frac{2N+1}{y_{j+N} - y_{j-N}} \tag{1}$$

where N is an even number (see fig. 1).

Calculating the Surface Gradients

In order to use cubic polynomials as the basis for interpolations on triangular distributions of known data, surface gradient estimates are needed at each data point. Gradient estimates at known points are based on linear triangles from the finite-element method using three nearby bounding points from the known data, Z_i. A surface within a triangular element can be described by a plane that retains a value of z within the bounding triangle:

$$z = \sum_{i=1}^{3} N_i(x,y)Z_i, \tag{2}$$

where z is a planar surface defined over any arbitrary triangle, $\Omega^{(I)}$ of known points, the Z_i's are known data values at the vertices or nodes of a triangle, and the $N_i(x,y)$'s are the so-called shape functions (see fig. 2). The index i denotes the first, second, and third points found in the search process (to be discussed in "Phase 2: Isolating the Three Bounding Points") that make up the bounding triangle. The shape functions, N_i, have the property of retaining a value of unity at the i'th node and zero at the other nodes in the triangle. This property ensures that $z = Z_i$ at the i'th node (Zienkiewics, 1979).

The shape functions used for simple triangular elements are defined mathematically as:

$$N_i(x,y) = (a_i + b_i x + c_i y)/2\Delta \qquad i = 1,2,3 \tag{3}$$

where i represents the permutation of the i'th node and v represents the area of the triangular element. The a_i, b_i, and c_i's are constant over the element and are calculated, strictly from the geometry of the triangle, using the following relations:

$$a_1 = x_2y_3 - x_3y_2, \tag{4a}$$

$$b_1 = y_2 - y_3, \tag{4b}$$

$$c_1 = x_3 - x_2. \tag{4c}$$

where (x_1,y_1) represents the spatial location of node "1" and the other coefficients (such as a_2, b_2, and c_2) are found by a cyclic permutation of the subscripts. The 2Δ in equation 3 is given by:

$$2\Delta = \begin{vmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{vmatrix}. \tag{5}$$

Finally, the gradients of this planar triangular surface can be determined by:

$$\frac{\partial z}{\partial x} = \sum_{i=1}^{3} \frac{\partial N_i}{\partial x}(x,y)Z_i \qquad \text{and} \qquad \frac{\partial z}{\partial y} = \sum_{i=1}^{3} \frac{\partial N_i}{\partial y}(x,y)Z_i \tag{6}$$

where:

$$\frac{\partial N_i}{\partial x}(x,y) = b_i/2\Delta \qquad \text{and} \qquad \frac{\partial N_i}{\partial y}(x,y) = c_i/2\Delta \qquad i=1,2,3. \tag{7}$$

Point density relations

$$\frac{\partial J}{\partial y}(J) = \frac{2N}{y_{2N} - y_1}; \qquad j = 1, 2N$$

$$\frac{\partial J}{\partial y}(J) = \frac{2N+1}{y_{j+N} - y_{j-N}}; \qquad j = (2N+1), (J_{max} - 2N - 1)$$

$$\frac{\partial J}{\partial y}(J) = \frac{2N}{y_{J_{max}} - y_{(J_{max}-2N), J_{max}}}; \qquad j = (J_{max} - 2N), J_{max}$$

Example calculation for N=3

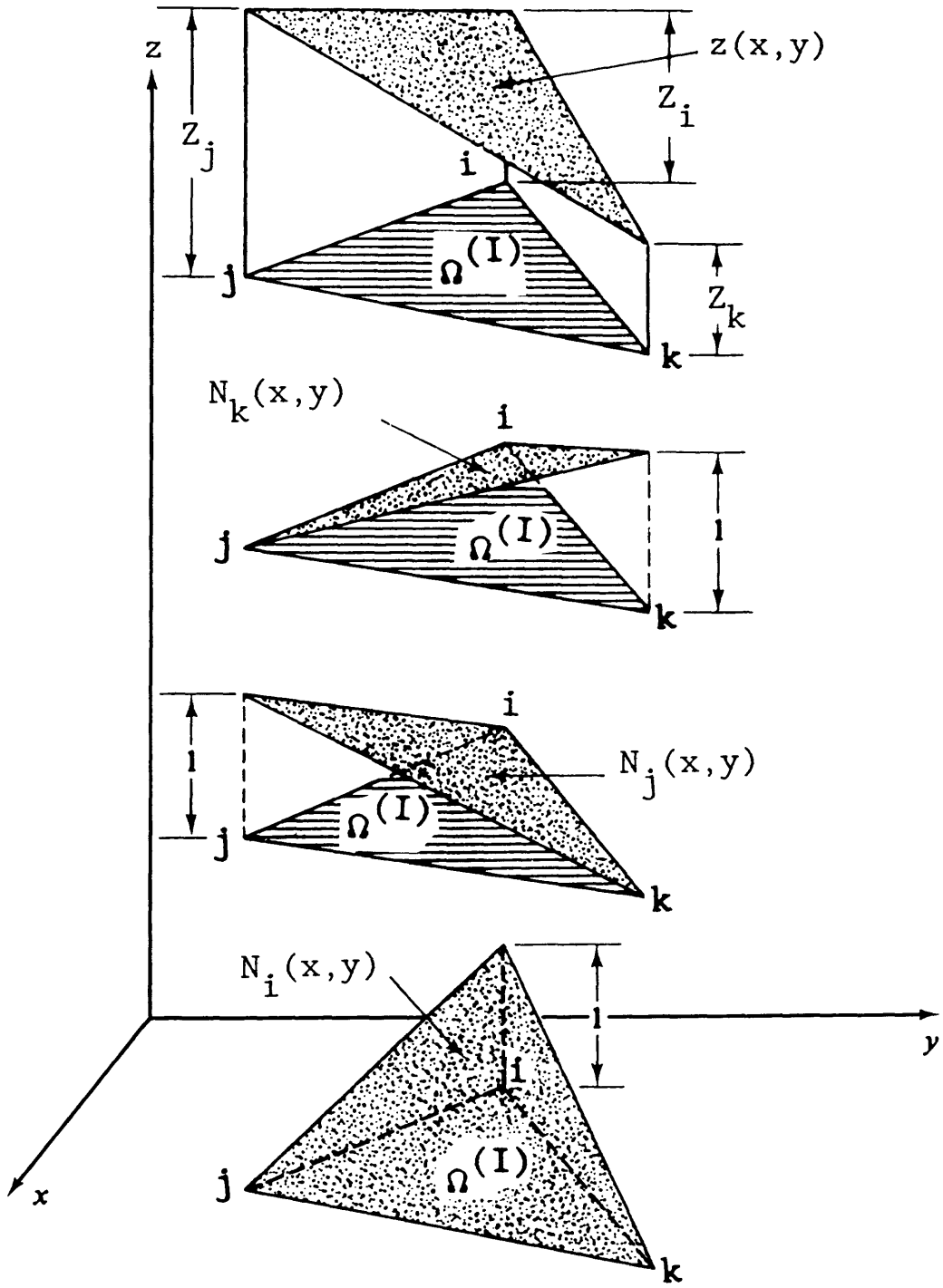
Sorted input data file

J	x	y	z
1	0.0	1.0	2.0
2	1.0	1.2	1.0
3	-3.1	1.3	0.0
.	.	.	.
.	.	.	.
j-N	9.1	5.4	2.0
.	10.5	5.5	2.1
.	5.5	5.7	3.7
j	-0.3	6.0	0.0
.	0.0	6.2	1.0
.	1.2	6.4	5.2
j+N	11.0	7.0	-0.1
.	.	.	.
.	.	.	.
J	0.2	10.4	0.9

$$\frac{\partial J}{\partial y}(J) = \frac{2N+1}{y_{j+N} - y_{j-N}}$$

$$= \frac{7}{7.0 - 5.4} = 4.375$$

FIGURE 1. Example of calculation of point density, $\frac{\partial J}{\partial y}(J)$.



$$z(x,y) = Z_i N_i(x,y) + Z_j N_j(x,y) + Z_k N_k(x,y)$$

FIGURE 2. Linear shape functions N_i over a triangular element $\Omega^{(I)}$ and their linear combination.

GENERATING BATHYMETRIC DATA

The basic task in generating bathymetric data is to interpolate depths at any arbitrary location from known data. The completion of this task is a two-step procedure: (1) The appropriate data on which the interpolations will be based must be isolated from the overall data base, and (2) interpolations need to be performed at user-specified (arbitrary) locations. The following sections outline the fundamentals of these procedures.

Location of Bounding Points

This method allows the data to be entered in a spatially random fashion (many data bases rely on regular orthogonal spacing of the data), which requires algorithms that search for a given subset of the data on which the interpolations are based. The interpolation method used in this report derives its interpolations from three nearby bounding data points of x^*,y^*, where x^*,y^* denotes the desired interpolation location. The algorithm that defines these three points from the overall data base uses a two-phase procedure. The first phase of the location algorithm isolates a subset of the overall data base in the general neighborhood of x^*,y^*. The second phase performs the final isolation of the three points used in the interpolation from the subset obtained in the first phase.

Phase 1: Finding the Neighborhood of the Bounding Points

In the preprocessing program, the data base is sorted according to the magnitude of y(J) to create a monotonically increasing relationship between y and J, and in addition, data point densities, $\frac{\partial J}{\partial y}(J)$, are calculated and stored for each data point in the data base. When the data is sorted in ascending value of the y-coordinate, a strip taken parallel to the x-axis, as shown in figure 3, corresponds directly to a contiguous section in the data file. An efficient search algorithm that takes advantage of these properties locates the neighborhood of the bounding points, using the first two terms in a Taylor series expansion about the data base index "J."

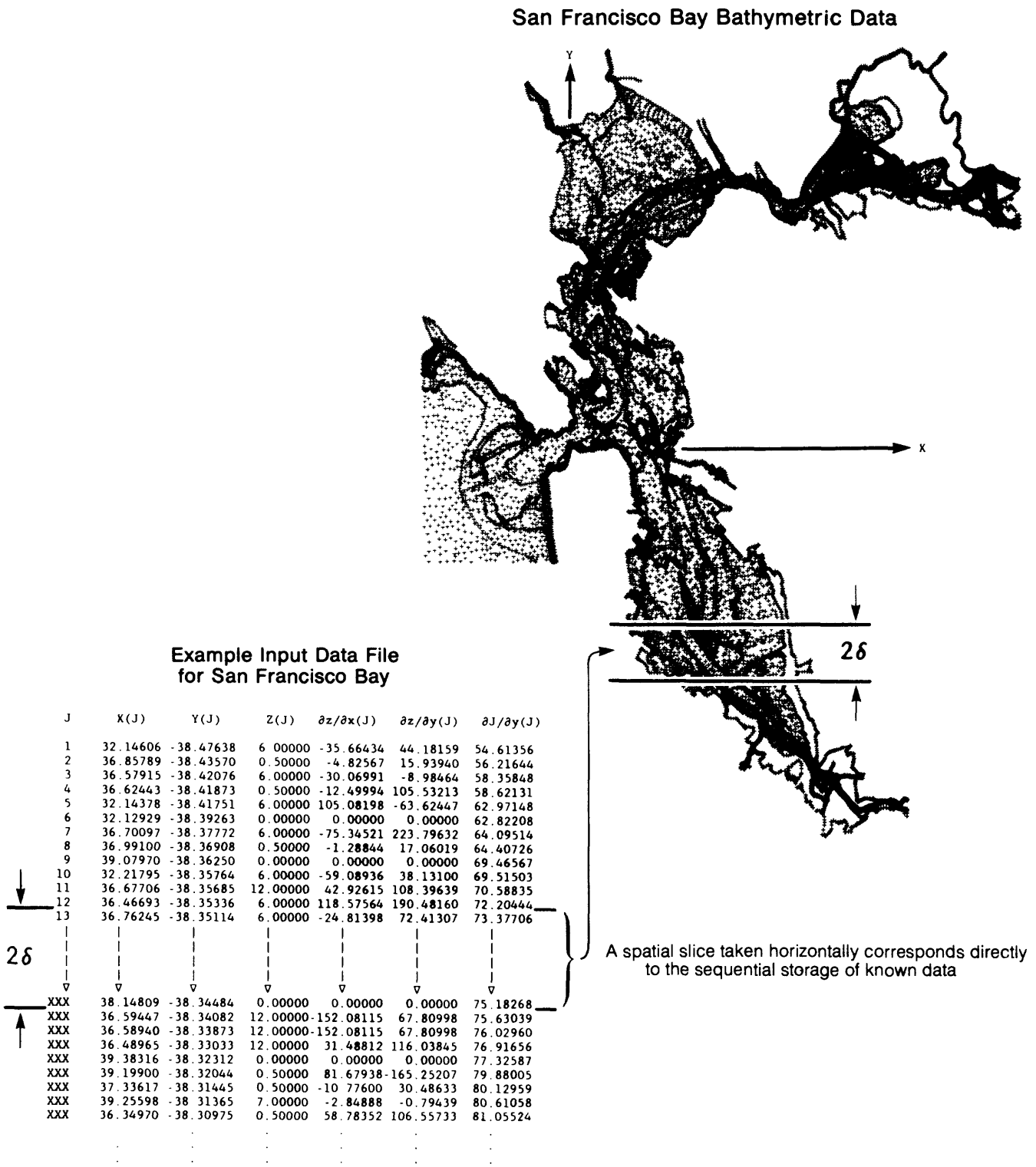


FIGURE 3. Example of contiguous relation between sorted data and spatial location.

Let x_J, y_J denote any point in the data base. The first two terms in a Taylor series applied to the data base pointer, J, can be written as:

$$J^* = J + [y^* - y_J]\frac{\partial J}{\partial y}(J), \tag{8}$$

which relates a spatial distance $[y^* - y_J]$ to a corresponding change in the pointer, J. For a given y^*, J^* is not known. Equation 8 can be used to estimate the data base pointer, J^*. An approximate J^* is the index of a known data point such that y_J^* is "close" to y^*. This form of the Taylor series is of little computational use because $\frac{\partial J}{\partial y}(J)$ varies within the data base with respect to J, or, alternatively, $\frac{\partial J}{\partial y}(J)$ depends on the value of J. Thus, an approximate value of J^* given by equation 8 may not be sufficiently accurate. This observation suggests an iterative method based on equation 8 to make successive iterations to locate J^*. A recurrence relation for finding an index value, J, within a user-specified neighborhood δ of y^* can be given as:

$$J^n = J^{n-1} + [y^* - y_{(J^{n-1})}]\frac{\partial J}{\partial y}(J^{n-1}) \qquad n=2,3,4\ldots, \tag{9}$$

where the superscript n indicates the order of iteration in the recurrence relation. The search for J^* is complete when a J^n in equation 9 is found as $|y^* - Y_J^n| \leq \delta$. To start the iteration sequence, an initial guess, n=1, is needed. In this method, the initial guess is estimated by:

$$j^1 = [y^* - y_1](\frac{\partial J}{\partial y})_{ave} \tag{10}$$

where y_1 is the first (or lowest y-value) known data point in the sorted data base, and $(\frac{\partial J}{\partial y})_{ave}$ is the average value of the point density taken over the entire data base calculated by:

$$(\frac{\partial J}{\partial y})_{ave} = [y_{J_{max}} - y_1]/J_{max} \tag{11}$$

where J_{max} is the total number of data points in the data base. Once the pointer is placed within δ of y^*, all the data within a δ radius of y^* are placed in a subdata base, which is used as the basis for the second phase of the search algorithm.

Phase 2: Isolating the Three Bounding Points

Given the subdata base found in phase 1, the basic algorithm used to locate the three bounding points (denoted 1,2,3 in fig. 4*C*) about a given spatial location (denoted 0 in fig. 4) is given in Thompson and Johnson (1985) and is summarized, with some modification, in this section. If the following set of conditions are not met for a given x,y coordinate location, the computer code assigns a zero for the interpolated value.

First point

The first point selected from the data base is the nearest point to the interpolation location, x^*, y^*, found by searching sequentially within a δ radius. This criterion is shown in figure 4*A* where $\vec{r}_{01}$ represents the vector from point 0 to point 1.

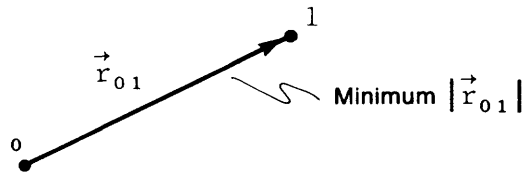
Second Point

The second point selected is the next closest point to 0 whose vector from point 0 forms an obtuse angle, ϕ ($\phi_{max} \geq \phi \geq \phi_{min}$), with the vector $\vec{r}_{01}$ where ϕ_{max} and ϕ_{min} are user specified (see fig. 4*B*). Thus, the second point must satisfy:

$$\cos(\phi_{max}) \leq \vec{u}_{01} \cdot \vec{u}_{02} \leq \cos(\phi_{min}), \tag{12}$$

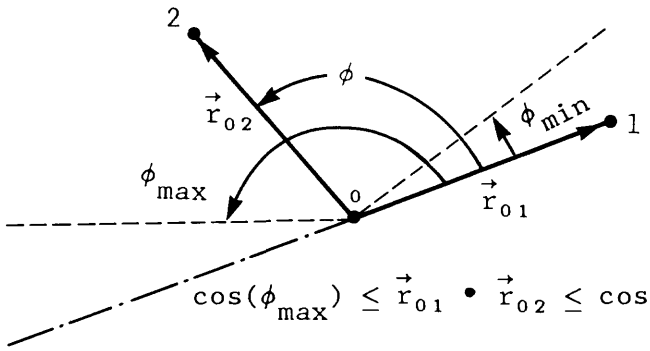
where the "$\cdot$" is the familiar inner or dot product of vectors, and $\vec{u}_{01}$, $\vec{u}_{02}$ are unit vectors in the $\vec{r}_{01}$, $\vec{r}_{02}$ directions, respectively. Values of ϕ_{min} and ϕ_{max} must be in the range of 0° to 180°. The terms "ϕ_{max}, ϕ_{min}" provide a means for specifying the admissible region for the second point. Restricting the admissible region of the second point ensures the bounding points selected by this algorithm will produce triangular regions with aspect ratios (height/base) on the order of one. In general, interpolation of surface gradients from a collection of points that have triangular regions which are nearly equilateral--aspect ratios roughly equal to one--produces better results. Erroneous gradients may be calculated from points that make up long narrow triangles.

A. First point



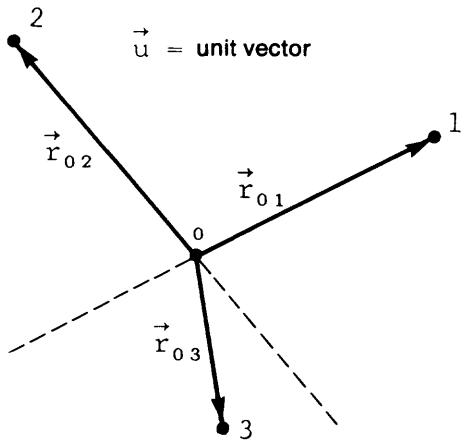
$$|\vec{r}_{01}| = \sqrt{(x_1-x_0)^2 - (y_1-y_0)^2}$$

B. Second point



$$\cos(\phi_{max}) \leq \vec{r}_{01} \bullet \vec{r}_{02} \leq \cos(\phi_{min})$$

C. Third point



$$|\vec{u}_{01}+\vec{u}_{02}+\vec{u}_{03}| \leq 1$$

FIGURE 4. Criteria for points that make up bounding triangle. "o" represents the interpolation location.

Third Point

The third point is the closest data point that lies within the zone created by the backward extensions of $\vec{r}_{01}$ and $\vec{r}_{02}$ in figure 4C. The third point will lie in the designated zone if

$$\vec{u}_{03} \cdot (-\vec{u}) \geq (-\vec{u}_{02}) \cdot (-\vec{u}), \tag{13}$$

where $\vec{u}_{03}$ represents the unit vector between points 0 and 3, and

$$\vec{u} = \frac{\vec{u}_{01} + \vec{u}_{02}}{\left|\vec{u}_{01} + \vec{u}_{02}\right|} \tag{14}$$

is the unit vector along the angular bisector between $\vec{r}_{01}$ and $\vec{r}_{02}$. The vertical bars in equation 14 represent the vector magnitude or Euclidian norm. When equation 14 is substituted for $\vec{r}$ in equation 13 and the identity

$$(\vec{u}_{01}+\vec{u}_{02}+\vec{u}_{03})^2 = 3 + 2(\vec{u}_{01} \cdot \vec{u}_{02} + \vec{u}_{02} \cdot \vec{u}_{03} + \vec{u}_{03} \cdot \vec{u}_{01}) \tag{15}$$

is applied, equation 13 becomes:

$$1 + [1/2(\vec{u}_{01}+\vec{u}_{02}+\vec{u}_{03})^2 + 3/2] \leq 0. \tag{16}$$

Or, finally the third point must satisfy,

$$\left|\vec{u}_{01}+\vec{u}_{02}+\vec{u}_{03}\right| \leq 1. \tag{17}$$

Notice that a restriction on the admissible region for point 2 effectively reduces the admissible region for point 3.

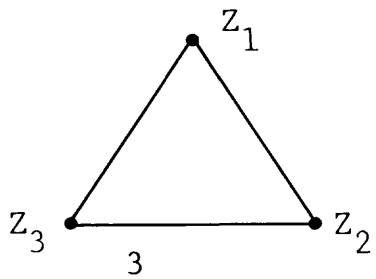
Interpolation Algorithms

Two different interpolation algorithms have been considered. One uses linear interpolation; the other, cubic polynomial interpolation (see fig. 5). Both are based on data found through the methodology of the preceding search sections.

Linear Triangles - C^0 Continuity

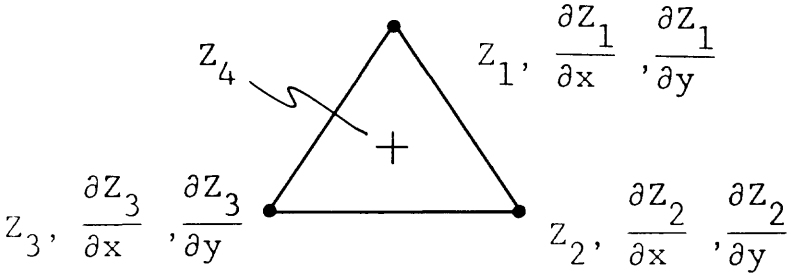
Equation 2 describes a planar surface within a triangular element. The plane described retains a value of Z_i at each interpolation point. Details of the mathematics behind this interpolation are given in the section, "Calculating the Surface Gradients."

Linear triangular element - C^0 continuity



$$z = \sum_{i=1}^{3} N_i(x,y) Z_i$$

Cubic triangular element - C^1 continuity



$$z = Z_4\phi_4 + \sum_{j=1}^{3} Z_j\phi_j +$$

$$\sum_{j=1}^{3} \left[\frac{\partial Z_j}{\partial x}(\phi_{xj})\frac{\partial x}{\partial N_1} + \frac{\partial Z_j}{\partial x}(\phi_{yj})\frac{\partial x}{\partial N_2} + \frac{\partial Z_j}{\partial y}(\phi_{xj})\frac{\partial y}{\partial N_1} + \frac{\partial Z_j}{\partial y}(\phi_{yj})\frac{\partial y}{\partial N_2}\right]$$

FIGURE 5. Example of interpolation relations based on three known points.

Cubic Triangles - C^1 Continuity

One approach to higher order interpolation using triangles is through the application of cubic polynomials. Using cubic polynomials allows for higher order interpolation by providing C^1 continuity: continuous mapping of the surface z across adjacent triangle boundaries as well as continuous first-order derivatives. The shape functions are derived using a cubic polynomial expansion (Lapidus and Pinder, 1982):

$$\phi_i = a + bN_1 + cN_2 + dN_1^2 + eN_2^2 + fN_1N_2 + gN_1^2N_2 + hN_1N_2^2 + iN_1^3 + jN_2^3, \quad (18)$$

which involves 10° of freedom represented by the 10 coefficients a through j; the N's are the shape functions introduced for the linear triangles. When Z_i, $\frac{\partial Z_i}{\partial x}$, $\frac{\partial Z_i}{\partial y}$ are specified at the corner nodes, the system is not closed; 1° of freedom is still needed. Normally, the last degree of freedom for this type of element is accounted for by applying a known value of Z at the centroid of the triangle. Obviously, given the expected randomness of the data used in this method, it is unlikely there will be a known data point at the centroid of every collection of three points in a spatially random data base. Therefore, another method must be used to obtain the last degree of freedom. Fortunately, by using the Z_i and its gradients at the corner nodes, a second-order accurate estimate can be made for Z at the centroid. At each node, a plane can be defined by Z_i, $\frac{\partial Z_i}{\partial x}$ and $\frac{\partial Z_i}{\partial y}$, where $\frac{\partial Z_i}{\partial x}$ and $\frac{\partial Z_i}{\partial y}$ are calculated for each data point from nearby data in the preprocessing step. (See the section, "Calculating the Surface Gradients" for details.) Each plane can be used to estimate the value of Z at the centroid. By exploiting the properties of the centroid, a simple arithmetic average of these nodal estimates is used to close the cubic polynomial. Mathematically, Z at the centroid x_c, y_c, denoted Z_c, is estimated by:

$$Z_c = 1/3 \sum_{i=1}^{3} [Z_i + \frac{\partial Z_i}{\partial x}(x_c - x_i) + \frac{\partial Z_i}{\partial y}(y_c - y_i)]. \quad (19)$$

With the system closed using Z_c, the functional mapping of the surface using cubic polynomials is:

$$z = Z_c\phi_4 + \sum_{j=1}^{3} Z_j\phi_j + \sum_{j=1}^{3} [\frac{\partial Z_j}{\partial x}(\phi_{xj})\frac{\partial x}{\partial N_1} + \frac{\partial Z_j}{\partial x}(\phi_{yj})\frac{\partial x}{\partial N_2} + \frac{\partial Z_j}{\partial y}(\phi_{xj})\frac{\partial y}{\partial N_1} + \frac{\partial Z_j}{\partial y}(\phi_{yj})\frac{\partial y}{\partial N_2}], \quad (20)$$

Where the shape functions using cubic polynomials are:

Node 1:

$$\phi_1 = N_1^2(N_1 + 3N_2 + 3N_3) - 7N_1N_2N_3 \tag{21}$$

$$\phi_{x1} = N_1^2(c_3N_2 - c_2N_3) + (c_2 - c_3)N_1N_2N_3 \tag{22}$$

$$\phi_{y1} = N_1^2(b_2N_3 - b_3N_2) + (b_3 - b_2)N_1N_2N_3 \tag{23}$$

Node 2:

$$\phi_2 = N_2^2(N_2 + 3N_3 + 3N_1) - 7N_1N_2N_3 \tag{24}$$

$$\phi_{x2} = N_2^2(c_1N_3 - c_3N_1) + (c_3 - c_1)N_1N_2N_3 \tag{25}$$

$$\phi_{y2} = N_2^2(b_3N_1 - b_1N_3) + (b_1 - b_3)N_1N_2N_3 \tag{26}$$

Node 3:

$$\phi_3 = N_3^2(N_3 + 3N_1 + 3N_2) - 7N_1N_2N_3 \tag{27}$$

$$\phi_{x3} = N_3^2(c_2N_1 - c_1N_2) + (c_1 - c_2)N_1N_2N_3 \tag{28}$$

$$\phi_{y3} = N_3^2(b_1N_2 - b_2N_1) + (b_2 - b_1)N_1N_2N_3 \tag{29}$$

Node 4:

$$\phi_4 = 27N_1N_2N_3 \tag{30}$$

The N_i, b_i, and c_i are defined in the section, "Calculating the Surface Gradients." The ϕ_i's retain a value of unity at each node i, and the ϕ_{xi}, ϕ_{yi} retain values of unit slope in the x and y directions, respectively (fig. 6).

DESCRIPTION AND OPERATION OF COMPUTER PROGRAMS

In the following sections, the details of program operation are given. Because the output from these programs is used as input to other computer codes, it is helpful to know the exact output formats so that an efficient transfer of information can be made between programs. Thus, in order to describe explicitly the input and output requirements, the input and output sections of the computer code are given in this document along with detailed descriptions of how they work.

To increase the efficiency of the search algorithms, the data are run through a preprocessing program (PREGRID.F77), which principally sorts the data and performs

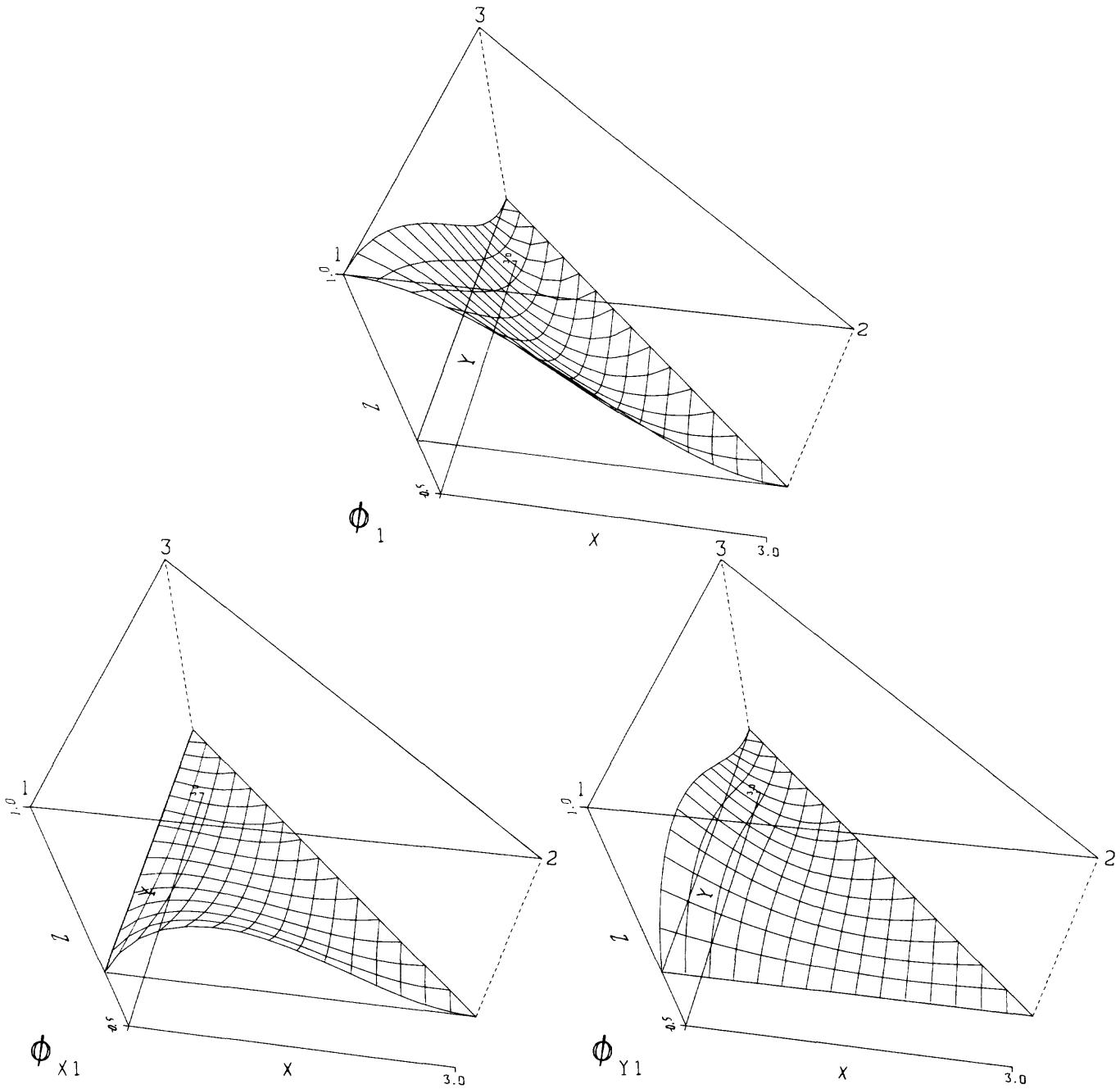


FIGURE 6. – Three-dimensional perspective plots of three cubic shape functions associated with node '1'.

calculations of associated properties. Additionally, the preprocessing program allows the data base to be edited so that, for example, changes in bathymetry can easily be incorporated into the existing data structure.

The interpolation program, GRID.F77, has three options designed specifically to generate depths for the computational networks of hydrodynamic models. The first option returns depths for finite-element networks. The second option calculates depths on regular orthogonal grids consistent with finite-difference methods. By taking advantage of the regularity of the grid in the second option, the depths are summed and used to estimate volumes. The final option calculates depths and areas along cross sections.

The computer codes for this method were designed as general, transportable, stand-alone programs. All input/output data used by the program are standard ASCII files. Both computer programs are written in standard FORTRAN 77 programming language and were developed and tested on PRIME 850 and 9955(II) computers. Flow charts and subroutine descriptions of both GRID.F77 and PREGRID.F77 are given in Appendix A.

The program dimensions are specified by the use of $INCLUDE (or $INSERT) files or equivalent. $INCLUDE files enable the COMMON blocks in all subroutines to be automatically dimensioned by specifying the dimensions on the COMMON blocks in the $INCLUDE file(s). $INCLUDE files allow for quick modification of the computer code to accommodate a wide variety of problem specifications. For both programs, the dimensions of all variables in the $INCLUDE files must be dimensioned to values greater than or equal to the number of known data points in the data base.

PROGRAM OPERATION - PREGRID.F77

To create a data base or to add, delete, or replace data in a pre-existing data base, two lines of control parameters must be included along with the data (if necessary) in a file called "ADD.DEPTH". An example of this file is given below:

```
------NBND----ANGMIN----ANGMAX------MAXT-------PER----IPOINT
       100       50.0      180.0          3       0.5        50
-----IEDIT-----XORIG-----YORIG------XLEN------YLEN-------PHI
         1       10.0        5.0      12.0        6.0       13.0
--------------X---------Y---------Z
        10.0800   10.0000    7.9996
        10.1600   10.0000    7.9984
        10.2400   10.0000    7.9964
        10.3200   10.0000    7.9936
        10.4000   10.0000    7.9900
        10.4800   10.0000    7.9856
```

A header line indicating the variable name and placement is inserted prior to each control line. The header lines are not read by the program but are necessary for the program to run properly. The program, as delivered, assumes the depths or z-values are in feet, and the x-values and y-values are in kilometers. The variables in this control file are defined as follows:

NBND = Twice the maximum number of points considered when searching for the second and third points from the first point.

ANGMIN = ϕ_{min} = Minimum allowable angle for selection of the second point. (See the section, "Isolating the Three Bounding Points" for details.)

ANGMAX = ϕ_{max} = Maximum allowable angle for selection of the second point. (See the section, "Isolating the Three Bounding Points" for details.)

MAXT = Maximum number of attempts made at finding a high aspect ratio triangle of bounding points, 0<MAXT<10. If a high aspect triangle is not found in the specified number of tries, the interpolations are based on the largest aspect triangle within the NBND search radius.

PER = Minimum acceptable aspect ratio for the triangle of bounding points in the gradient calculation. For an equilateral triangle the aspect ratio is 1.0. A typical value for this variable is 0.5.

IPOINT = ["N" in point density calculation] = Number of points used to calculate point densities.

IEDIT = Flag that denotes the type of changes to be made in the data set to be made:

If IEDIT = 1: the data which are input through the file "ADD.DEPTH" REPLACES the existing data in the rectangular area defined by the following input quantities.

If IEDIT = 0: the data which are input through the file "ADD.DEPTH" are appended to the pre-existing data set without any deletions.

(XORIG,YORIG) = X_o,Y_o (fig. 7) The global coordinates of the lower left-hand grid point.

(XLEN,YLEN) = L_x,L_y (fig. 7) The lengths (kilometers) of the grid coordinate axes in the horizontal and vertical directions, respectively.

PHI = ϕ (fig. 7) The angle the finite-difference grid makes with the global coordinate system.

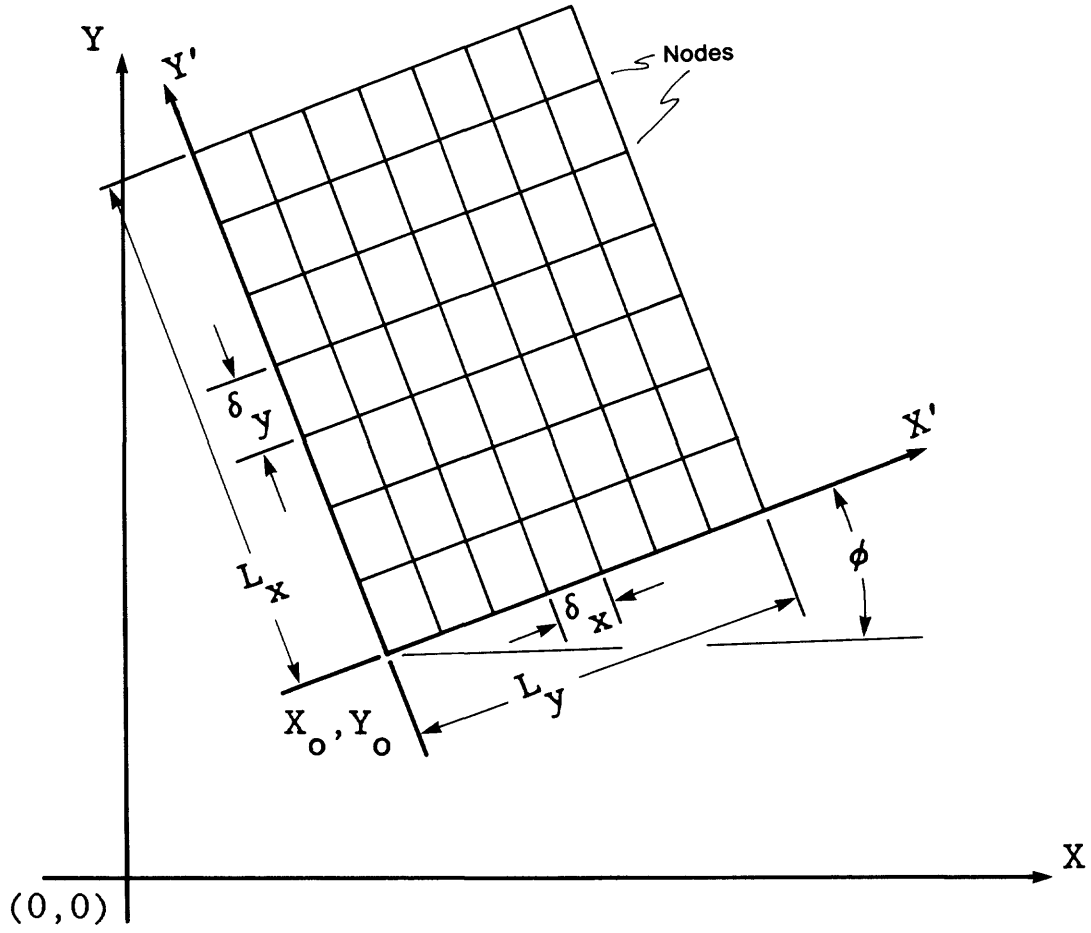


FIGURE 7. Definition of input variables for finite-difference option in PREGRID.F77.

The following FORTRAN statements are used to read in these data:

```
      READ(5,231)                                  NBND,ANGMIN,ANGMAX,MAXT,PER,IPOINT
231   FORMAT(/I10,2F10.5,I10,F10.5,I10/)
      READ(5,233)                                     IEDIT,XORIG,YORIG,XLEN,YLEN,PHI
233   FORMAT(I10,5F10.5/)
      READ(5,5,END=6)                                                    X(N),Y(N),Z(N)
5     FORMAT(5X,8F10.5)
```

The output from PREGRID.F77 is put in a file called "DEPTH.DATA.NEW." In order to use the interpolation program GRID.F77, the name of this file must be changed from "DEPTH.DATA.NEW" to "DEPTH.DATA." Output from the PREGRID.F77 is accomplished through the following FORTRAN statements:

```
      WRITE(10,5) X(I),Y(I),Z(I),PARX(I),PARY(I),D
5     FORMAT(5X,8F10.5)
```

which puts the bathymetric data in the following form:

```
  37.36411 -38.23727   6.00000 -38.18353  37.66636 113.40565
  31.76653 -38.23658   0.00000   0.00000   0.00000 127.01021
  37.55679 -38.22966   6.00000 -24.35769  73.21034 127.62857
  36.21996 -38.22842   6.00000   0.00000   0.00000 128.91650
  39.35606 -38.22591   0.00000   0.00000   0.00000 128.73544
    .        .         .         .         .         .
    .        .         .         .         .         .
    .        .         .         .         .         .
```

Where PARX(I),PARY(I) are the calculated surface gradients (see the section "Calculating the Surface Gradients") and "D" is the point density for the i'th point (see the section "Calculating the Point Densities").

When running the preprocessing program, IPOINT ("N" in equations 1, 2, and 3) must be specified as input by the user. Large values of IPOINT will mask local fluctuations in $\frac{\partial J}{\partial y}(J)$, providing a smooth functional relation between $\frac{\partial J}{\partial y}(J)$ and J. Because each successive iteration for the pointer location in the Taylor series search algorithm is based on $\frac{\partial J}{\partial y}(J)$, a highly variable $\frac{\partial J}{\partial y}(J)$ could lead to an oscillatory convergence, or, in the worst case, to no convergence at all. Therefore, some smoothing of $\frac{\partial J}{\partial y}(J)$ is desirable as long as it does not impact the larger scale trend in $\frac{\partial J}{\partial y}(J)$. The large scale variations in $\frac{\partial J}{\partial y}(J)$ are what drive the convergence of this method; thus, an appropriate value ($\cong$1 percent of the total number of data points) of IPOINT should be selected to match each data base.

PROGRAM OPERATION - GRID.F77

The first part of the program reads the user's data requests, opens files, and reads the depth data. The program then begins the two-phase search process and subsequently performs the interpolation for each requested x^*,y^* location.

To run the interpolation program GRID.F77, a control file called GRID.INPUT must be created that contains the input and output file names and the other operational parameters needed for each run of the program. The following is an example GRID.INPUT file:

```
  --METHOD---METER---INTER----NBND------RMAX----ANGMIN----ANGMAX
         1       1       0     100       1.0      50.0     180.0
-----XORIG-----YORIG-----XSPAC-----YSPAC-----XGRID-----YGRID-----PHI
        -2.892    14.840      24.5      23.5      0.25      0.25       0.0
-----------------------------------OUTPUT FILENAME
SANPAB.DEPTH
```

As with ADD.DEPTH, a header line precedes each new data line. The header statement is intended to aid in the readability of this file and to be used as a template when generating new data sets. To use the template, right-justify the data entries to the last letter in the variable name of the header statement. The program skips these headers but depends on their placement to run properly. The input variables are defined as follows:

METHOD = Interpolation option flag:
If METHOD=1; Interpolate a finite-difference grid.
If METHOD=2; Interpolate at x,y coordinates contained in a user specified file.
If METHOD=3; Interpolate on cross sections.

METER = Output units flag:
If METER=1; Depths are output in feet.
If METER=2; Depths are output in meters.

INTER = Interpolation flag:
If INTER=0; Linear interpolation.
If INTER=1; Cubic interpolation.

NBND = Two times the maximum number of points considered when searching for the second and third points from the first point.

RMAX = The radius that defines the maximum distance within which points will be considered for interpolation. Interpolations outside this distance are considered zero.

ANGMIN = ϕ_{min} = Minimum allowable angle for selection of the second point. (See the section, "Isolating the Three Bounding Points" for details.)

ANGMAX = ϕ_{max} = Maximum allowable angle for selection of the second point. (See the section, "Isolating the Three Bounding Points" for details.)

The top line of data is required for all interpolation options. In this example, the finite-difference option (interpolation on regularly spaced orthogonal grids) is used. The remaining variables and parameters in this example are considered in the next sections where the specifics of the interpolation options are discussed.

Finite-Difference Grids

The spatial attributes of a given finite-difference grid can be defined by the global coordinate location of its lower left-hand grid point, X_o,Y_o, the length of the grid in the horizontal and vertical directions, L_x,L_x, the incremental spacing of the sampling points, δ_x, δ_y, and by the angle, ϕ, the grid makes with the global coordinate system (fig. 7). The computer code transforms a coordinate system based on the user-specified finite-difference grid into the global coordinates using the following translation and rotation relations:

$$x^*_{i,j} = x_o + (i-1)\delta_x\cos\phi - (j-1)\delta_y\sin\phi \quad i = 1,2,..I_{max}, \quad j = 1,2,..J_{max} \tag{31}$$

$$y^*_{i,j} = y_o + (i-1)\delta_y\cos\phi + (j-1)\delta_x\sin\phi \quad i = 1,2,..I_{max}, \quad j = 1,2,..J_{max} \tag{32}$$

where

$$I_{max} = INT(L_x/\delta_x) + 1, \text{ and} \tag{33}$$

$$J_{max} = INT(L_y/\delta_y) + 1. \tag{34}$$

Volumes of the area covered by a given grid are calculated by summing over the entire grid the product of the average of the depths at the corners of a cell and the area of the cell. Mathematically this can be expressed as:

$$Vol = (\delta_x\delta_y/4) \sum_{i=1}^{(I_{max}-1)} \sum_{j=1}^{(J_{max}-1)} [d_{i,j} + d_{i+1,j} + d_{i,j+1} + d_{i+1,j+1}], \tag{35}$$

where $d_{i,j}$ is the interpolated depth at the mesh point i,j. By rearranging the summations, a computationally more efficient form of the volume computation can be given as:

$$Vol = \delta_x\delta_y[V_1/4 + V_2/2 + V_3], \tag{36}$$

where:

$$V_1 = d_{1,1} + d_{I_{max},1} + d_{1,J_{max}} + d_{I_{max},J_{max}} \tag{37}$$

$$V2 = \sum_{i=2}^{(I_{max}-1)} [d_{i,1} + d_{i,J_{max}}] + \sum_{j=2}^{(J_{max}-1)} [d_{1,j} + d_{I_{max},j}], \quad (38)$$

$$V3 = \sum_{i=2}^{(I_{max}-1)} \sum_{j=2}^{(J_{max}-1)} d_{i,j}. \quad (39)$$

The volume of a given grid is output to a file named "VOL.OUT."

Input Requirements

The input requirements for the generation of finite-difference grids essentially include the parameters that control the search sequence as previously discussed, the quantities in figure 7 that relate the finite-difference grid-coordinate system to the global coordinates, and, finally, the output filename. The following is an example control file (GRID.INPUT) for the finite-difference option:

```
-METHOD---METER---INTER-----NBND------RMAX----ANGMIN----ANGMAX
        1       1       0     100       1.0      50.0     180.0
-----XORIG-----YORIG-----XSPAC-----YSPAC-----XGRID-----YGRID-------PHI
       -2.892    14.840      24.5      23.5      0.25      0.25       0.0
-----------------------------------OUTPUT FILENAME
SANPAB.DEPTH
```

where:

(XORIG,YORIG) = X_o, Y_o (fig. 7) The global coordinates of the lower left-hand grid point.

(XSPAC,YSPAC) = L_x, L_y (fig. 7) The lengths (kilometers) of the grid-coordinate axes in the horizontal and vertical directions, respectively.

(XGRID,YGRID) = δ_x, δ_y (fig. 7) The grid spacing (kilometers) in the horizontal and vertical grid-coordinate directions, respectively.

PHI = ϕ (fig. 7) The angle the finite-difference grid makes with the global coordinate system.

Using the San Francisco data set shown in figure 8 as an example, a three-dimensional perspective warped surface of the finite-difference mesh generated using the control file given above for San Pablo Bay is shown in figure 9. The following read statements are used to input the data for the finite-difference option:

```
    READ(64,601) METHOD,METER,INTER,NBND,RMAX,ANGMIN,ANGMAX
601 FORMAT(/4I8,3F10.5)
    READ(64,602) XORIG,YORIG,XSPAC,YSPAC,XGRID,YGRID,PHI
602 FORMAT(/8F10.5)
    READ(64,661) JFILE
661 FORMAT(/A50)
```

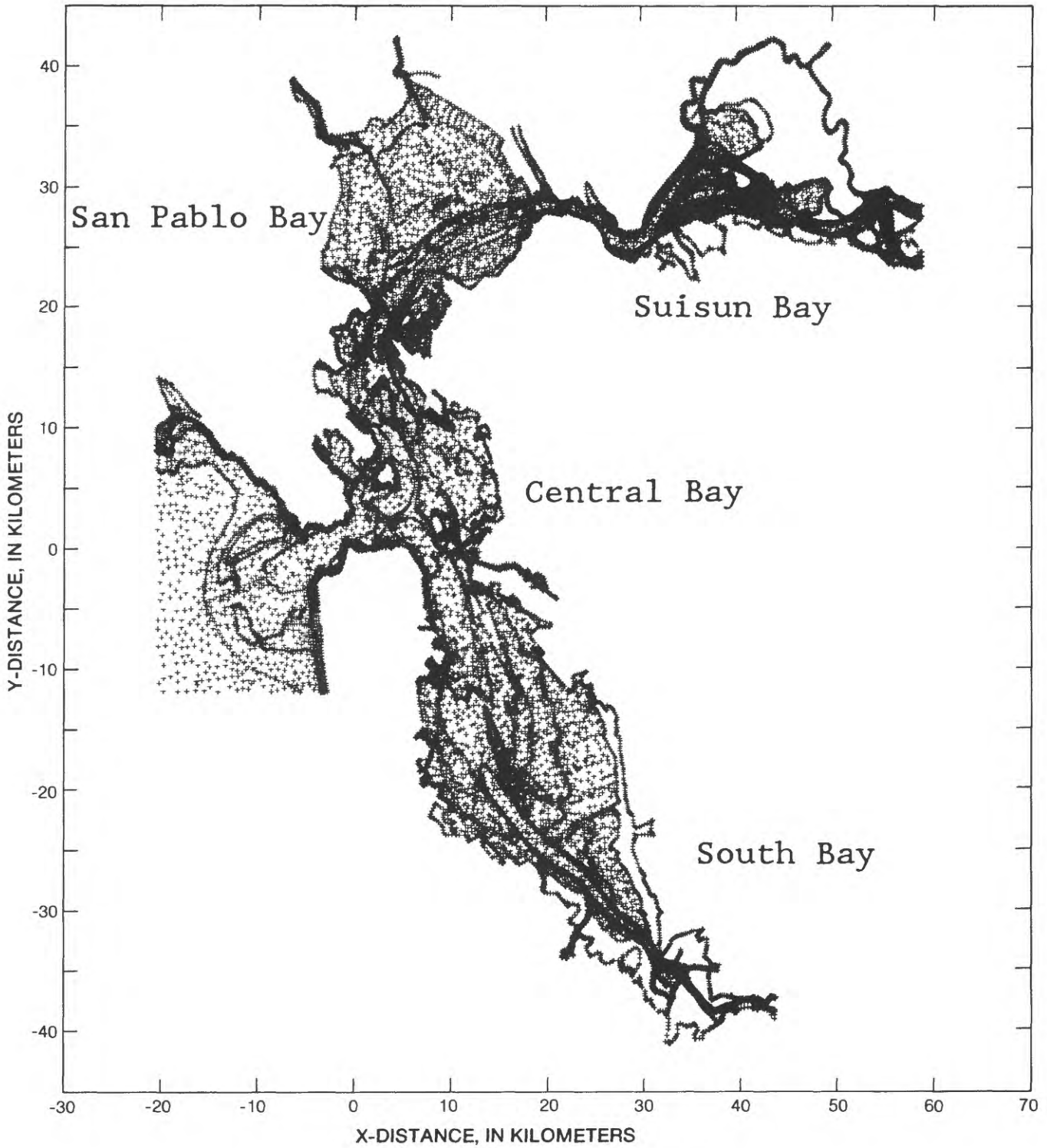


FIGURE 8. Spatial extent and variability of bathymetric data base. Each '+' represents a location of known depth.

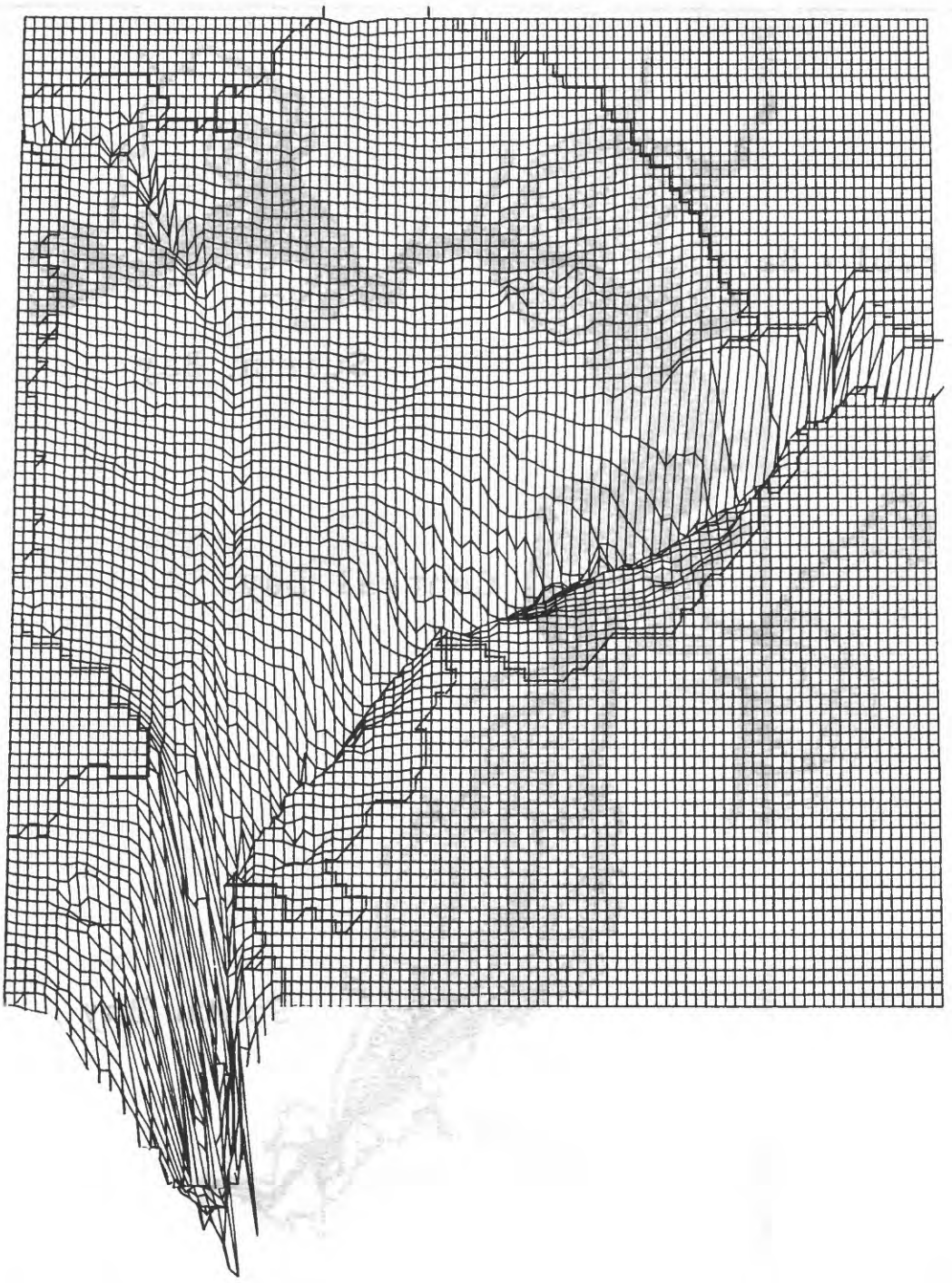


FIGURE 9. – San Pablo Bay finite-difference grid shown in relief.

Output Format

Interpolated depths from the finite-difference grid generation option are provided in a matrix of depths, $d_{i,j}$. The output is in integers with the decimal place moved to the right by one digit; that is, the output depths = INT(10.0*$d_{i,j}$). This format saves storage by not writing out all the decimal points. In general, the entire $d_{i,j}$ matrix of interpolated depths will not fit on an 80-column page width; therefore, $d_{i,j}$ is output in increments of j=10 (fig. 10). Thus, the first 10 j values are output for all i, i=1,I_{max} and then the j values, $11 \leq j \leq 20$ are output and so on until all the j values have been output for all i. The following FORTRAN code is used to accomplish this output format:

```
    NR1 = 1
    NR2 = 10
C
    IX = IFIX(XSPAC/XGRID)
    IY = IFIX(YSPAC/YGRID)
    II=0 = 0
732 CONTINUE
    IF(NR2.GT.(IY+1)) NR2 = IY+1
    DO 631 I = 1,IX+1
        KK = 0
        II = II + 1
        DO 632 J = NR1,NR2
            KK = KK + 1
C
C...Calculation of depths occurs in this block
C
632     CONTINUE
C
        WRITE(55,912)(NINT(ZMAT(II,KP)*10.0),KP=1,KK)
912     FORMAT(10I5)
C
631 CONTINUE
C
    NR1 = NR1 + 1
    NR2 = NR2 + 10
    IF(NR1.LE.(IY+1)) GO TO 732
```

Where:

IX+1 = I_{max} = The total number of mesh points in the finite-difference grid in the horizontal direction,

IY+1 = J_{max} = The total number of mesh points in the finite-difference grid in the vertical direction, and

ZMAT = $d_{i,j}$ = The matrix of the interpolated depths.

Interpolations for Finite-Element Grids

Unlike the finite-difference calculations, the spatial location of the nodes that make up finite-element networks do not, in general, follow any regular pattern. Thus, to interpolate depths using the finite-element option, the user must supply a file which contains the spatial location of each node in the network in the coordinate system on which the known data was collected.

Input Requirements

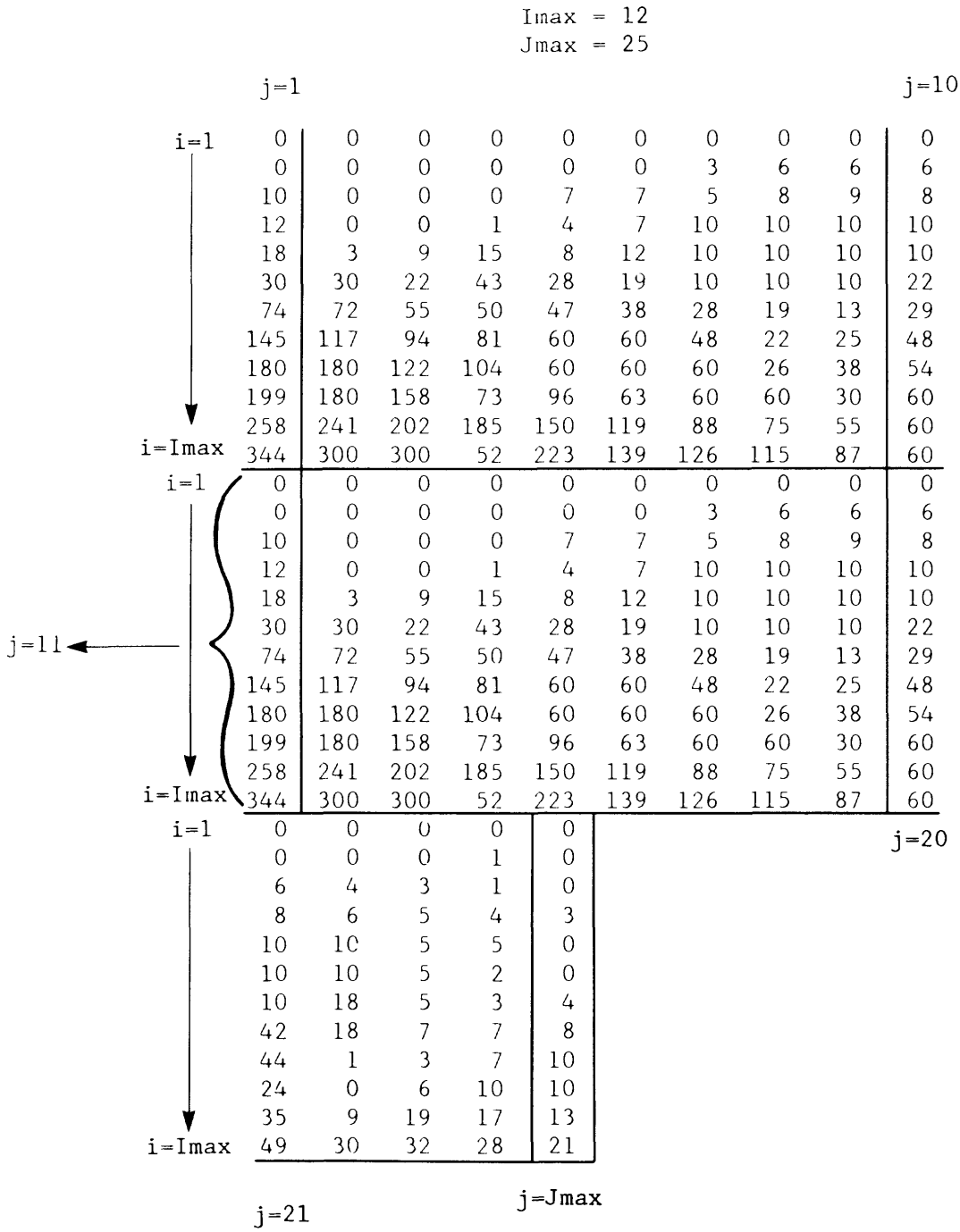
The control file for this option contains the interpolation parameters (the first line of the control file), the name of the input file that contains the coordinates of the computation points where the interpolations are desired, and the output file name which will contain the interpolated depths along with the coordinates at each desired interpolation point. An example control file has this form:

```
--METHOD---METER---INTER----NBND------RMAX----ANGMIN----ANGMAX
       2       1       0     100       1.0      50.0     180.0
-----------------------------------INPUT FILENAME
FINITE.E.INPUT
-----------------------------------OUTPUT FILENAME
FINITE.E.OUTPUT
```

The file that contains the input coordinates is read using the following FORTRAN code:

```
622 READ(13,620,END=621) XF,YF
620 FORMAT(3F10.3)
    GO TO 622
621 CONTINUE
```

Example
Finite-difference output format



0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	3	6	6	6
10	0	0	0	7	7	5	8	9	8
12	0	0	1	4	7	10	10	10	10
18	3	9	15	8	12	10	10	10	10
30	30	22	43	28	19	10	10	10	22
74	72	55	50	47	38	28	19	13	29
145	117	94	81	60	60	48	22	25	48
180	180	122	104	60	60	60	26	38	54
199	180	158	73	96	63	60	60	30	60
258	241	202	185	150	119	88	75	55	60
344	300	300	52	223	139	126	115	87	60
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	3	6	6	6
10	0	0	0	7	7	5	8	9	8
12	0	0	1	4	7	10	10	10	10
18	3	9	15	8	12	10	10	10	10
30	30	22	43	28	19	10	10	10	22
74	72	55	50	47	38	28	19	13	29
145	117	94	81	60	60	48	22	25	48
180	180	122	104	60	60	60	26	38	54
199	180	158	73	96	63	60	60	30	60
258	241	202	185	150	119	88	75	55	60
344	300	300	52	223	139	126	115	87	60
0	0	0	0	0					
0	0	0	1	0					
6	4	3	1	0					
8	6	5	4	3					
10	10	5	5	0					
10	10	5	2	0					
10	18	5	3	4					
42	18	7	7	8					
44	1	3	7	10					
24	0	6	10	10					
35	9	19	17	13					
49	30	32	28	21					

FIGURE 10. Example output for finite-difference option. Interpolations for a (12 x 25) finite-difference grid.

Where XF,YF are the coordinates of the location where the interpolation is desired. The input file takes on the following form:

```
 13.943   -10.231
  9.123   100.345
  2.987   -20.980
 -2.937    22.567
   .         .
   .         .
   .         .
```

Output Format

The file that contains the interpolated depths for the finite-element option is produced using the following FORTRAN code:

```
    WRITE(55,620) XF,YF,ZZ
620 FORMAT(3F10.3)
```

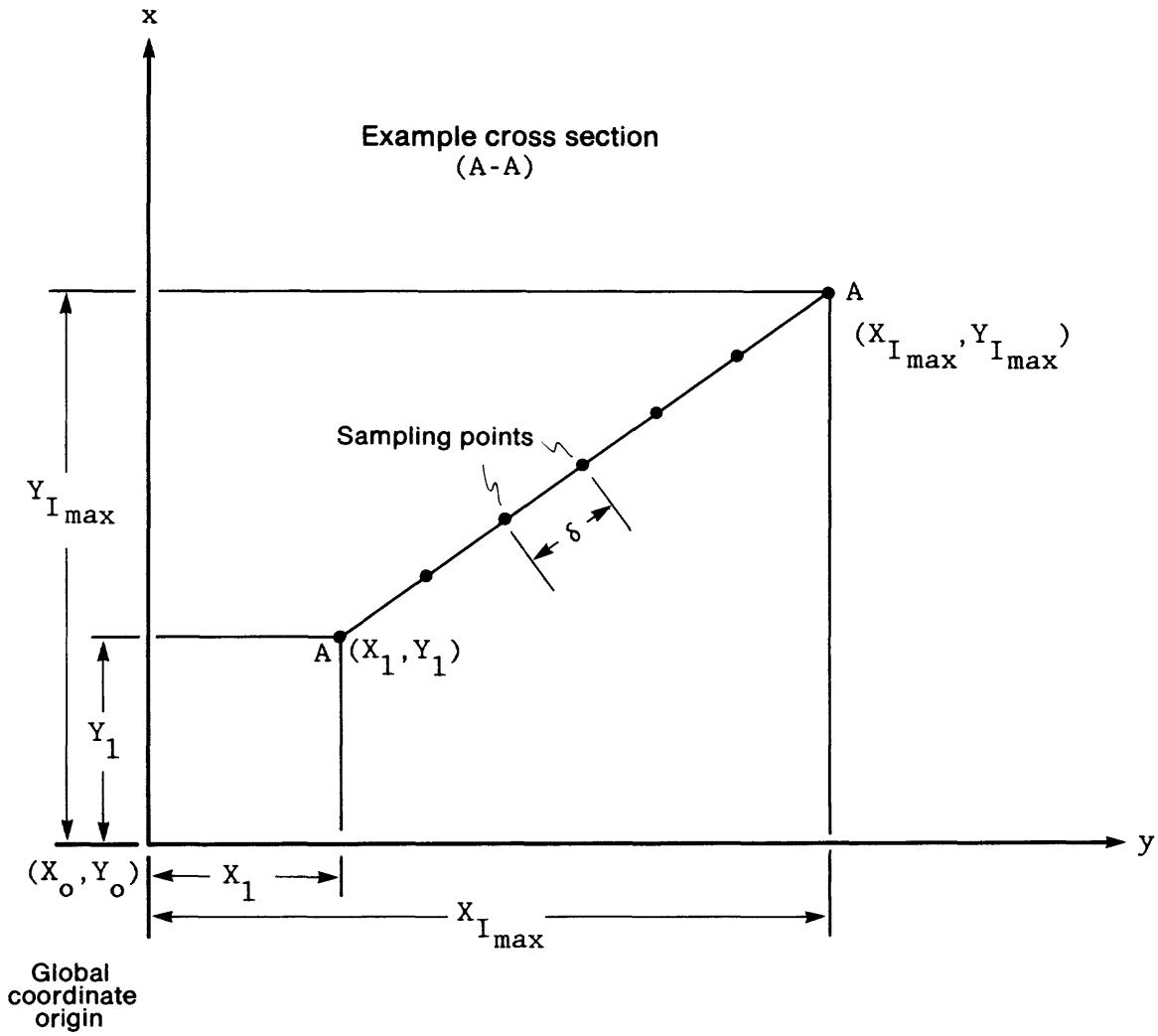
which produces the following output:

```
 13.943   -10.231     0.000
  9.123   100.345     2.985
  2.987   -20.980    10.312
 -2.937    22.567     1.768
   .         .
   .         .
   .         .
```

where ZZ is the interpolated depth.

Cross Sections

In the study of flow problems there often is a need to look at cross-sectional information. Points on a cross section can be uniquely defined by the coordinates of the cross-section endpoints X_1,Y_1 and $X_{I_{max}}, Y_{I_{max}}$ and a constant spacing interval δ (fig. 11). Cross sections can be determined between any two arbitrary endpoints at any orientation.



$$\delta = \sqrt{(X_{I_{max}} - X_1)^2 + (Y_{I_{max}} - Y_1)^2 / (I_{max} - 1)}$$

FIGURE 11. Definition of terms used in cross-section option.

Because constant interval sampling is used to determine cross sections, the trapezoidal rule is used for the calculation of the cross-sectional area, A:

$$A = \delta/2[d_1 + d_{I_{max}}] + \delta[\sum_{i=1}^{(I_{max}-1)} d_i]. \quad (40)$$

Because the total cross-sectional area can be expressed as the product of the average depth and the top width of the section, the average depth can be calculated as:

$$\bar{d} = A/\sqrt{(X_{I_{max}} - X_1)^2 + (Y_{I_{max}} - Y_1)^2} \quad (41)$$

where the denominator represents the cross-section length or top width.

Input Requirements

To run the cross-section option, the user must provide in the control file the interpolation parameters (given in the first line of the control file), the input and output file names, and the number of sampling points, NGRID.

```
--METHOD---METER---INTER----NBND------RMAX----ANGMIN----ANGMAX
       3       1       0     100       1.0      50.0     180.0
-----------------------------------INPUT FILENAME
SAMPLE_XSEC.INPUT
----------------------------------OUTPUT FILENAME
SAMPLE_XSEC.OUTPUT
---NGRID
     100
```

In the input file, the user specifies a cross-section designation or name and the endpoints of the cross section for each cross section desired (a maximum of 50 cross sections can be run in a single program run).

```
-------XSECTION NAME-----XSTRT-----YSTRT-----XSTOP-----YSTOP
  XSEC1                    -4.7     -0.5       1.7        8.5
  XSEC2                    -4.7     -0.5       9.0       -0.5
```

where:

(XSTRT,YSTRT) = X_1,Y_1 (fig. 11) = The first pair of end points defined in global coordinates.

(XSTOP,YSTOP) = $X_{I_{max}}$, $Y_{I_{max}}$ (fig. 11) = The second pair of end points defined in global coordinates.

NGRID = I_{max} = The number of sampling points (fig. 11).

Output Format

Output for the cross-section option includes, from left to right in the first line of the example shown below, the cross-section name, the starting location, XSTRT,YSTRT, and the ending location, XSTOP,YSTOP, of the cross-section line, the mean depth, the number of sampling points, and the cross-sectional area. After the first line, a sequence of lines are given each containing the coordinate location x,y, the interpolated depth, and the length along the cross section measured from XSRT,YSTRT for each sampling point along the cross section. An example cross section taken in San Francisco Bay from Pier 39 in San Francisco across Alcatraz and Angel Islands to the Tiburon Peninsula is shown in figure 12.

```
XSEC1                0.000     0.000     0.000    10.000 57.9  101  579.087
                     0.000000  0.000000  1.068722  0.000000
                     0.000000  0.100000  6.638384  0.100000
                     0.000000  0.200000 12.426405  0.200000
                     0.000000  0.300000 17.998901  0.300000
                     0.000000  0.400000 42.654488  0.400000
                     0.000000  0.500000 59.074371  0.500000
                     0.000000  0.600000 60.000084  0.600000
                     0.000000  0.700000 60.000183  0.700000
                     0.000000  0.800000 59.999123  0.800000
                      .         .         .         .
                      .         .         .         .
                      .         .         .         .
```

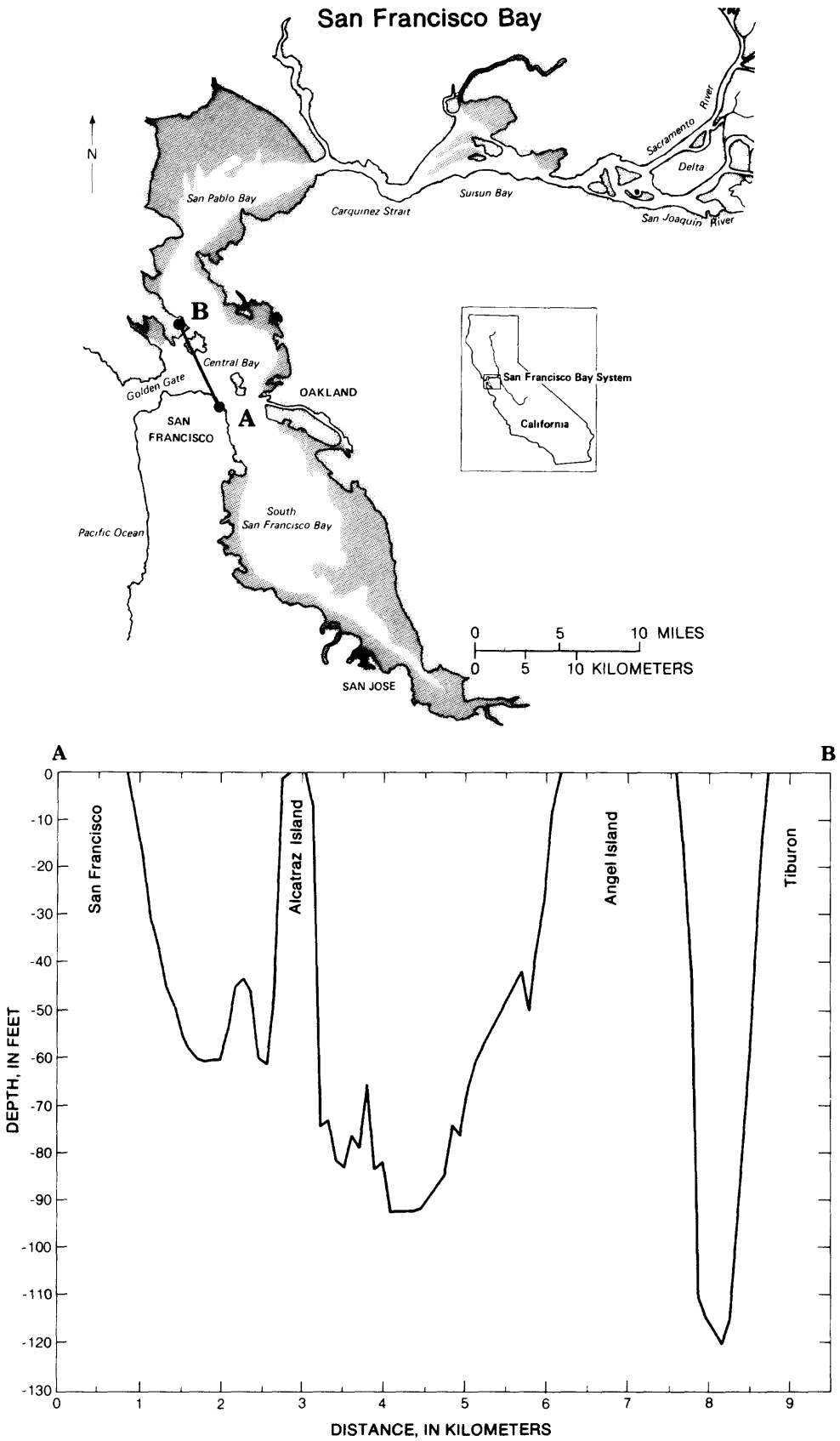


FIGURE 12. Example cross section through Central Bay.

EXAMPLE: SAN FRANCISCO BAY DATA SET

This bathymetric interpolation method has been extensively applied in the study of hydrodynamic processes in the San Francisco Bay estuary using a data base consisting of roughly 26,000 points of known depth (see fig. 8). All depths in the data base were obtained directly from the NOAA-NOS charts, measured in feet referenced to Mean Lower Low Water (MLLW). Roughly 80 percent of the data base consists of points that were taken from bathymetric contours on the NOAA-NOS charts; the remaining 20 percent consists of random points that fill in the gaps between contours. Each + in figure 8 represents a known depth position. Spatially, the depths are referenced to a coordinate system whose origin is centered on the Presidio tide station (lat 37°48'24", long 122°27'54") and whose coordinate axes are measured in kilometers east (positive X) and north (positive Y).

SUMMARY

An efficient methodology for creating the bathymetric data necessary for hydrodynamic numerical modeling studies is presented. The search algorithms, interpolation routines, and all necessary preprocessing procedures are discussed. Detailed computer program documentation and operational procedures are given, including example applications of the various program options to the San Francisco Bay estuary.

REFERENCES

Barnhill, R.E., 1977, Representation and approximation of surfaces, in Rice, J.R., editor, Mathematical software, volume 3: New York, Academic Press, p. 69-120.

Cole, J.W., 1978, ANSI FORTRAN IV, A structured programming approach: Dubuque, Iowa, Wm. C. Brown, 420 p,

Franke, Richard, and Nielson, Greg, 1980, Smooth interpolation of large sets of scattered data: International Journal for Numerical Methods in Engineering, 15, 1691, 1704.

Lapidus, Leon, and Pinder, G.F., 1982, Numerical solution of partial differential equations in science and engineering: New York, John Wiley, 677 p.

Thompson, J.F., and Johnson, B.H., 1985, Development of an adaptive boundary-fitted coordinate code for use in coastal and estuarine areas: U.S. Army Corps of Engineers Waterways Experiment Station, Vicksburg, Mississippi, Miscellaneous Paper HL-85-5.

Zienkiewics, O.C., 1979, The finite element method (3d ed.): New York, McGraw Hill, 787 p.

APPENDIX A

Program Flow Charts and Subroutine Descriptions

Depicted in figures 13, 14, and 15 are flow charts of the subroutine calls used in GRID.F77 and PREGRID.F77, which are described in greater detail here.

INPUT:
Reads in the data base of known depths, including:

a) x,y,z information for each known point.
b) The surface gradients $\frac{\partial z}{\partial x}$, $\frac{\partial z}{\partial y}$ and point density, $\frac{\partial J}{\partial y}(J)$, at each point.
c) Calculates the average point density $(\frac{\partial J}{\partial y})$ave.

FIND:
Places the data-base pointer within δ of the interpolation point using the Taylor series expansion technique. This routine first checks to see if the interpolation point falls within the limits of the known data. If the interpolation point is outside the limits of the known data, a value of zero is returned for the depth. If the data-base pointer is not within δ in 10 tries, a sequential search is used to locate an appropriate pointer location. Subroutine FIND calls POINTS, TRILIN, TRICUB (fig. 14).

POINTS:
Finds the closest three bounding points following a modified version of Thompson's algorithm. Subroutine POINTS calls SEARCH.

SEARCH:
Finds the closest point to the interpolation point. This routine skips points that are discarded by subroutine POINTS for not fitting the bounding criterion.

TRILIN:
Given three bounding points and the interpolation point, this routine returns a depth based on linear interpolation (interpolation based on a plane through the three points).

TRICUB:
Given three bounding points and the interpolation point, this routine returns a depth based on cubic polynomial interpolation. Subroutine TRICUB calls CENT and SHAPE.

CENT:
Given the x,y coordinates of the bounding triangle, this subroutine exploits the fact that the centroid of any triangle in local coordinates is 1/3,1/3 to calculate the global coordinates of the centroid of the bounding triangle.

SHAPE:
This subroutine calculates the value of the cubic polynomial shape functions given the coordinates of the interpolation point and the linear shape function values.

Preprocessing Specific Routines

GRAD:
Given three known bounding points, this subroutine calculates the gradients $\frac{\partial z}{\partial x}$, $\frac{\partial z}{\partial y}$ at the known data points based on linear triangular shape functions. When no bounding points are found, as is the case for data along the domain boundary, then a gradient of zero is returned (fig. 15).

EDIT:
This subroutine deletes all of the data within a user-specified rectangular region by evaluating the signs of the dot products between each data point and the four corners of the rectangle.

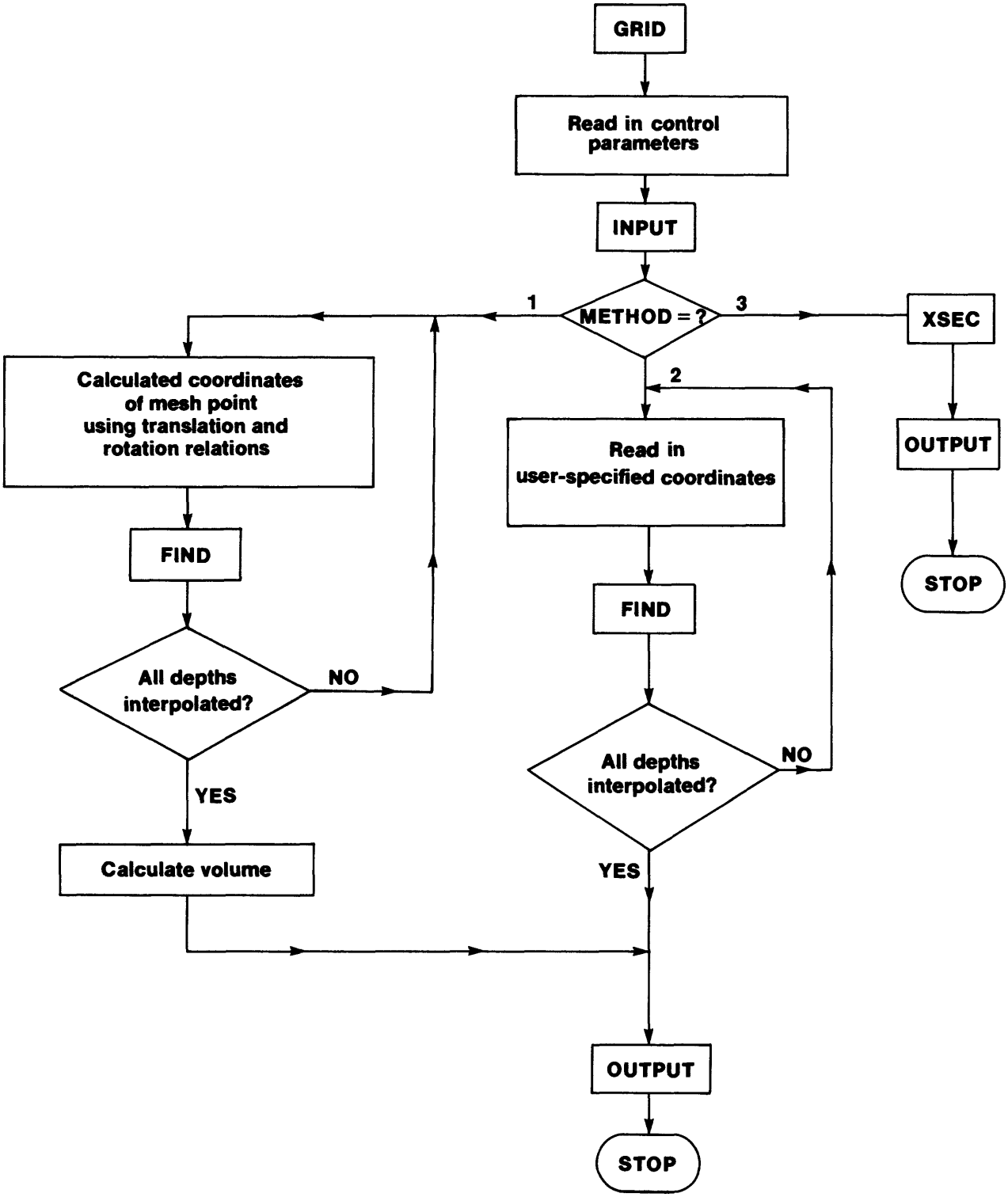


FIGURE 13. – Flow chart of main program.

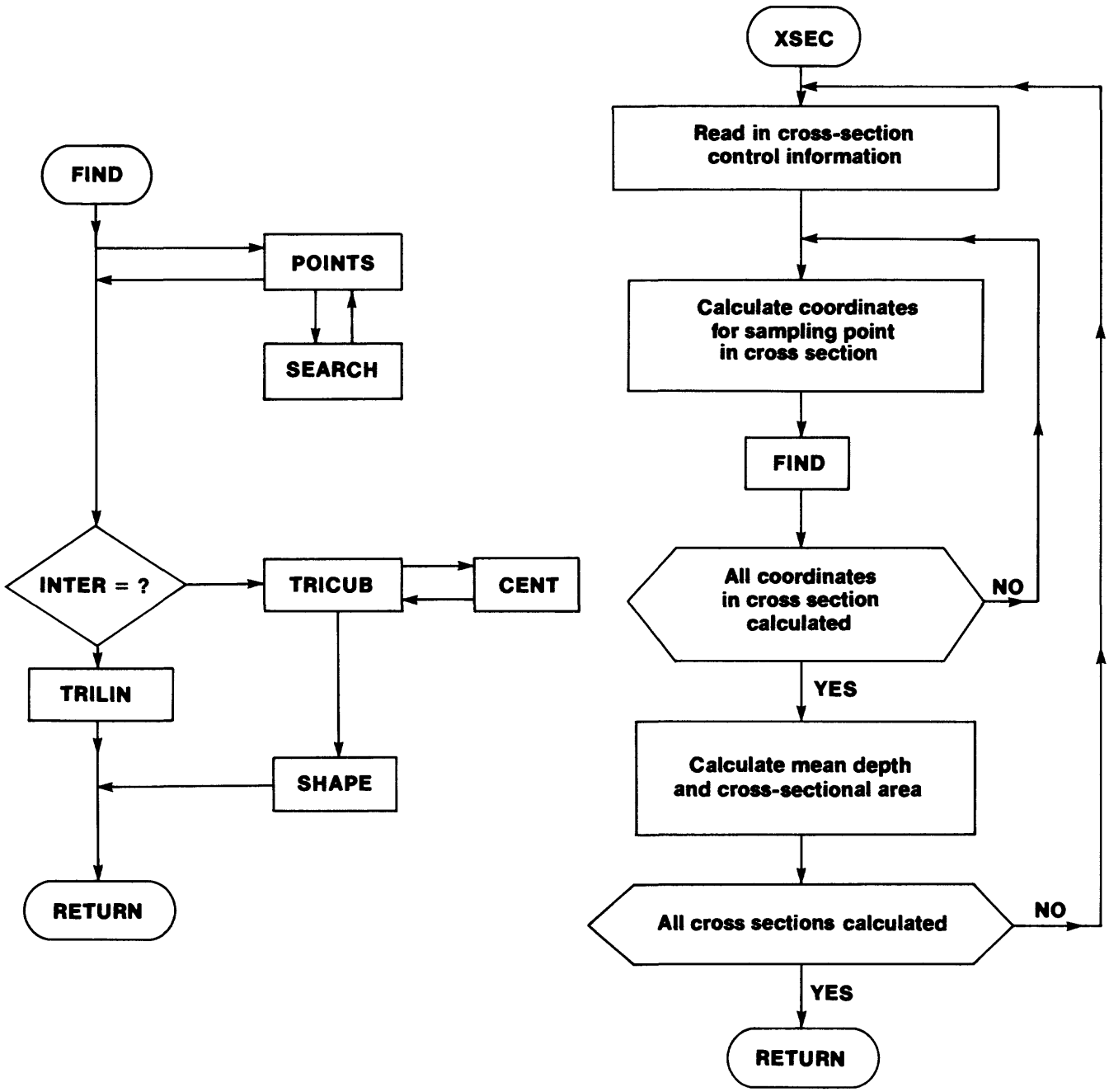


FIGURE 14. – Flow chart of subroutines **FIND** and **XSEC**.

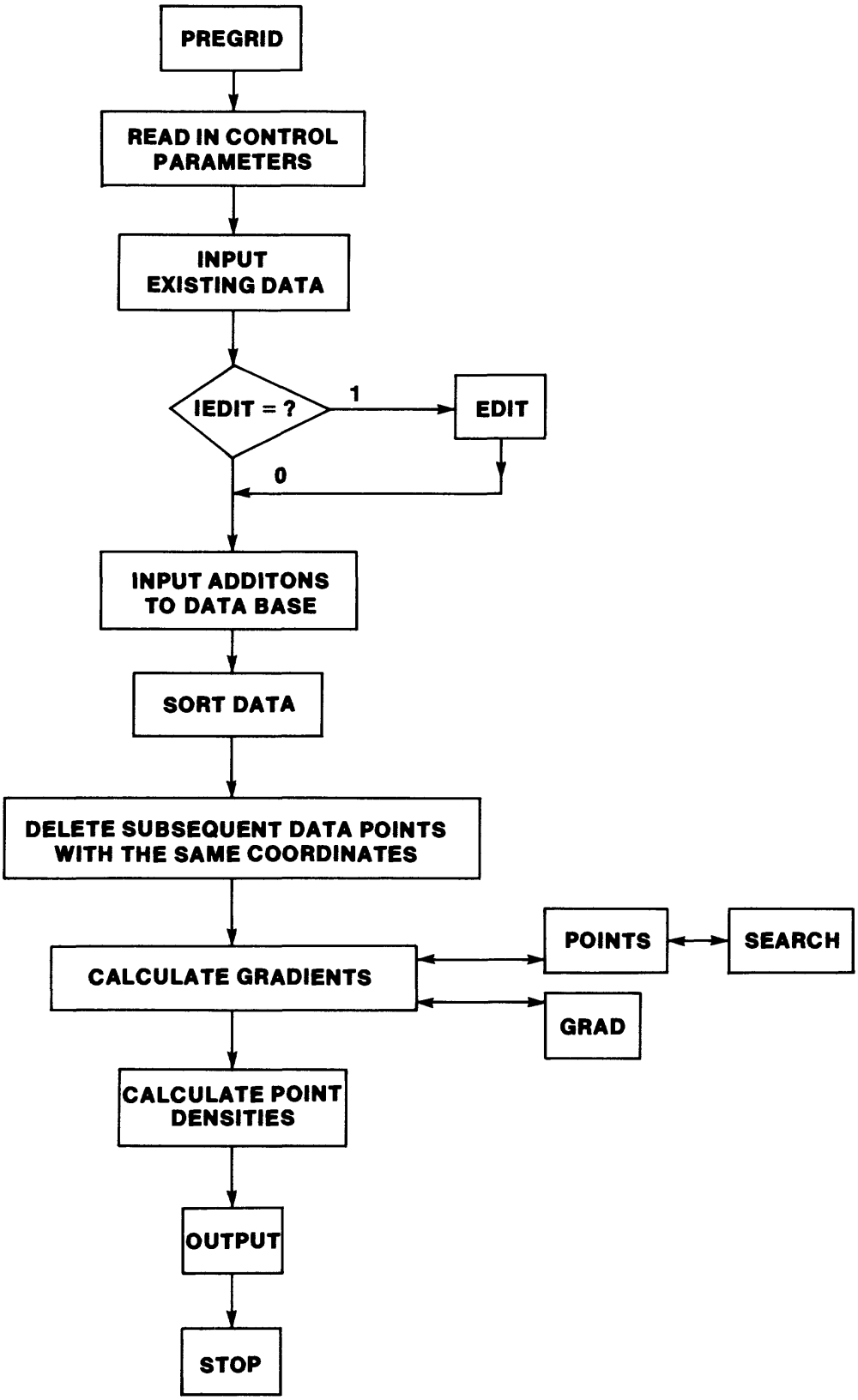


FIGURE 15. Flow chart of preprocessing program PREGRID.F77.

APPENDIX B

Variable List

Name	FORTRAN name	Array size	Type	Remarks
Data Base Variables				
	NMAX		INT	Number of known data points.
x	X	NMAX	REAL	X-coordinate location of known data.
y	Y	NMAX	REAL	Y-coordinate location of known data.
Z	Z	NMAX	REAL	Depth of known data.
$\frac{\partial J}{\partial y}(J)$	D	NMAX	REAL	Point density of known data.
$(\frac{\partial J}{\partial y})$ave	DENS	NMAX	REAL	Average point density.
$\frac{\partial z}{\partial x}$	PARX	NMAX	REAL	Surface gradient with respect to x.
$\frac{\partial z}{\partial y}$	PARY	NMAX	REAL	Surface gradient with respect to y.
Search Control Variables				
	METHOD		INT	Interpolation option flag: If METHOD=1, Interpolation for finite-difference grids; If METHOD=2, Interpolation at x,y coordinates contained in a user-specified file; If METHOD=3, Interpolation on cross sections.
	METER		INT	Output units flag: If METER=1, Depths output in feet; If METER=2, Depths output in meters.
	INTER		INT	Interpolation flag: If INTER=0, Linear; If INTER=1, Cubic.
	NBND		INT	Two times the maximum number of points considered when searching for the second and third points from the first point.

APPENDIX B--Continued

Name	FORTRAN name	Array size	Type	Remarks
δ	RMAX		REAL	The radius that defines the maximum distance within which points will be considered for interpolation. Interpolations outside this distance are considered zero.
ϕ_{min}	ANGMIN		REAL	Minimum allowable angle for selection of the second point. (See the section, "Isolating the Three Bounding Points" for details.)
ϕ_{max}	ANGMAX		REAL	Maximum allowable angle for selection of the second point. (See the section, "Isolating the Three Bounding Points" for details.)
	(CMIN,CMAX)		REAL	Cosines of ANGMIN and ANGMAX, respectively.
Search Variables				
	XF		REAL	X-coordinate location where interpolation is desired.
	YF		REAL	Y-coordinate location where interpolation is desired.
	ZF		REAL	Interpolated depth at XF,YF.
	IFIND1,IFIND2,IFIND3		REAL	The pointer locations that define the bounding triangle.
	RMIN		REAL	Keeps track of the minimum distance from the interpolation point to the first point in the bounding triangle.
	IEXCL	1000	INT	Stores the pointer number of each point that does not meet the bounding triangle requirements.
	NUMEX		INT	Counter: The total number of points that did not meet the bounding triangle requirements.
	XXMIN,XXMAX,YYMIN,YYMAX		REAL	Limits of known data.

Name	FORTRAN name	Array size	Type	Remarks
Finite-Difference Variables				
x_o, y_o	XORIG,YORIG		REAL	The global coordinates of the lower left-hand grid point.
L_x, L_y	XLEN,YLEN		REAL	The lengths of the grid coordinate axes in the horizontal and vertical directions of the finite-difference grid.
δ_x, δ_y	XGRID,YGRID		REAL	Grid spacings in the horizontal and vertical directions of the finite-difference grid.
ϕ	PHI		REAL	The angle the finite-difference grid makes with the global coordinate system.
	CPHI,SPHI		REAL	Cosine and sine of ϕ, respectively.
I_{max}	IX		INT	Number of finite-difference grid points in the X-direction.
J_{max}	IY		INT	Number of finite-difference grid points in the Y-direction.
	ZMAT	10	REAL	Temporary storage of finite-difference depths.
Preprocessing Variables				
N	IPOINT		INT	The number of points used to calculate point densities.
	IEDIT		INT	Flag that denotes the type of additions to be made: If IEDIT = 1, the added data REPLACES the existing data in the rectangular area defined by the parameters used to define the finite-difference grids. If IEDIT = 0, the added data is appended to the pre-existing data set without any deletions.

Name	FORTRAN name	Array size	Type	Remarks
	MAXT		REAL	The maximum number of attempts made at finding a high aspect ratio triangle of bounding points; an integer such that 0<MAXT<10.
	PER		REAL	Controls the minimum acceptable aspect ratio for the triangle of bounding points for the gradient calculation. For an equilateral triangle, the aspect ratio is 1.0. A typical value for this variable is 0.5.
Cross-Section Variables				
	NAME	50	CHAR	Cross-section name.
	X	50,201	REAL	Temporary storage for all X-coordinates of cross-section sampling points.
	Y	50,201	REAL	Temporary storage for all Y-coordinates of cross-section sampling points.
	Z	50,201	REAL	Temporary storage for all calculated depths at cross-section sampling points.
$\bar{d}$	DEPTH	50	REAL	Average depth for each cross section.
X_1, Y_1	X1,Y1	50	REAL	Starting end-point coordinates for cross section.
$X_{I_{max}}, Y_{I_{max}}$	X2,Y2	50	REAL	Ending end-point coordinates for cross section.
	DIS	50	REAL	Distance measured along the cross section.

Name	FORTRAN name	Array size	Type	Remarks
NGRID	K		INT	Number of sampling points along the cross section.
	MAX		INT	Maximum number of cross sections.
	MAXNUM		INT	Maximum number of allowable sampling points.
	XDIV		REAL	Sampling increment in the x-direction.
	YDIV		REAL	Sampling increment in the y-direction.
	XL		REAL	Total length of cross section.
	SLOPE		REAL	Slope of the cross-section line.
Interpolation Variables				
a_i, b_i, c_i	A,B,C	3	REAL	Linear shape function constants.
Δ	A1		REAL	Area of the triangular element.
Z_c	ZC		REAL	Estimated depth at centroid of element.
x_c, y_c	XBAR,YBAR		REAL	Coordinates of the centroid of the triangle.
N_i	L	3	REAL	Value of linear shape function at XF,YF.
ϕ_i	PH	4	REAL	Cubic shape function: equals one at i.
ϕ_{xi}	PHX	3	REAL	Cubic shape function: x-gradient equals one at i.
ϕ_{yi}	PHY	3	REAL	Cubic shape function: y-gradient equals one at i.