Automated Programming of Explicit Shallow-water Models Part I. Linearized Models with Linear or Quadratic Friction

R. F. Henry

Institute of Ocean Sciences
Department of Fisheries and Oceans
Sidney, B.C. V8L 4B2

1982

Canadian Technical Report of Hydrography and Ocean Sciences No. 3

Fisheries and Oceans
Pêches et Océans

Canada

Canadian Technical Report of Fisheries and Aquatic Sciences

Technical reports contain scientific and technical information that contributes to existing knowledge but which is not normally appropriate for primary literature. Technical reports are directed primarily toward a worldwide audience and have an international distribution. No restriction is placed on subject matter and the series reflects the broad interests and policies of the Department of Fisheries and Oceans, namely, fisheries and aquatic sciences.

Technical reports may be cited as full publications. The correct citation appears above the abstract of each report. Each report is abstracted in *Aquatic Sciences and Fisheries Abstracts* and indexed in the Department's annual index to scientific and technical publications.

Numbers 1-456 in this series were issued as Technical Reports of the Fisheries Research Board of Canada. Numbers 457-714 were issued as Department of the Environment, Fisheries and Marine Service, Research and Development Directorate Technical Reports. Numbers 715-924 were issued as Department of Fisheries and the Environment, Fisheries and Marine Service Technical Reports. The current series name was changed with report number 925.

Technical reports are produced regionally but are numbered nationally. Requests for individual reports will be filled by the establishment listed on the front cover and title page. Out-of-stock reports will be supplied for a fee by commercial agents.

Rapport technique canadien des sciences halieutiques et aquatiques

Les rapports techniques contiennent des renseignements scientifiques et techniques qui constituent une contribution aux connaissances actuelles, mais qui ne sont pas normalement appropriés pour la publication dans un journal scientifique. Les rapports techniques sont destinés essentiellement à un public international et ils sont distribués à cet échelon. Il n'y a aucune restriction quant au sujet; de fait, la série reflète la vaste gamme des intérêts et des politiques du ministère des Pêches et des Océans, c'est-à-dire les sciences halieutiques et aquatiques.

Les rapports techniques peuvent être cités comme des publications complètes. Le titre exact paraît au-dessus du résumé de chaque rapport. Les rapports techniques sont résumés dans la revue *Résumés des sceiences aquatiques et halieutiques*, et ils sont classés dans l'index annual des publication scientifiques et techniques du Ministère.

Les numéros 1à 456 de cette série ont été publiés à titre de rapports techniques de l'Office des recherches sur les pêcheries du Canada. Les numéros 457 à 714 sont parus à titre de rapports techniques de la Direction générale de la recherche et du développement, Service des pêches et de la mer, ministère de l'Environnement. Les numéros 715 à 924 ont été publiés à titre de rapports techniques du Service des pêches et de la mer, ministère des Pêches et de l'Environnement. Le nom actuel de la série a été établi lors de la parution du numéro 925.

Les rapports techniques sont produits à l'échelon régional, mais numérotés à l'échelon national. Les demandes de rapports seront satisfaites par l'établissement auteur dont le nom figure sur la couverture et la page du titre. Les rapports épuisés seront fournis contre rétribution par des agents commerciaux.

Canadian Technical Report of
Hydrography and Ocean Sciences 3

1982

AUTOMATED PROGRAMMING OF EXPLICIT SHALLOW-WATER MODELS

Part I. Linearized Models with Linear or Quadratic Friction

by

R.F. Henry

Institute of Ocean Sciences
Department of Fisheries and Oceans
Sidney, B.C. V8L 4B2

Cat. No. FS 97-18/3 ISSN 0711-6764

Correct citation for this publication:

Henry, R.F. 1982. Automated programming of explicit shallow-water models. Part I. Linearized models with linear or quadratic friction. Can. Tech. Rep. Hydrogr. Ocean Sci. 3 : iv + 70 p.

CONTENTS

ABSTRACT

Henry, R.F. 1982. Automated programming of explicit shallow-water models. Part I. Linearized models with linear or quadratic friction. Can. Tech. Rep. Hydrogr. Ocean Sci. 3 : iv + 70p.

This report describes a system for reducing the programming effort required to develop explicit finite-difference models of shallow-water phenomena. Information about the model geometry is reduced to a set of numerical codes, which are checked graphically. These codes are used during model runs to control the finite-difference calculations throughout the model. Use of standard predeveloped subroutines provided greatly reduces the likelihood of programming error and speeds up model development for a modest increase in computing costs.

Key words: programming, shallow-water, model, automated.

RÉSUMÉ

Le present rápport décrit un système qui permet de réduire l'effort de programmation nécessaire pour élaborer des modèles explicites de différences finies de phénomènes qui se produisent en eau peu profonde. L'information concernant la géométrie des modèles est réduite a une série de codes numériques, qui sont vérifiés graphiquement. Ces codes sont utilisés au cours des essais de modèles afin de contrôler, pour l'ensemble du modèle, les calculs des différences finies. L'utilisation des sous-programmes normalisés pré-développés réduit de beaucoup les possibilites d'erreur de programmation et accélère le développment de modèles au prix d'une légère augmentation des coûts de calcul.

Mots-clés: programmation, eau peu profonde, modèle, automatisé.

1. Introduction:

This report describes a method which simplifies numerical modelling of long-wave phenomena such as tides, tsunamis and storm surges. The governing shallow water equations are represented by means of a common explicit finite-difference scheme; the main innovation is the method proposed for automating programming of all the calculations required in the interior of the model and at the boundaries. The latter may include closed (land) boundaries, for which a zero normal transport condition is assumed, boundaries where waves radiate out of a model into adjacent water bodies, and boundaries at which the exchange with a river or adjacent sea can be specified as a function of time. Different procedures are required in each of these cases and all differ from the calculations carried out at interior points of the model.

The irregular geometry of most model domains complicates programming very severely. Normally at each time step, variables are computed row by row (or column by column) across a model grid. Unless a boundary happens to be parallel to a coordinate axis, the row (column) where the boundary is located varies from row to row (column to column) and the program must contain instructions identifying which grid meshes in each row (column) require special boundary calculations. But the same type of calculation is required at all boundary points of similar nature, irrespective of position on the grid. It follows that the problem of defining where a particular type of boundary calculation is required can be separated from that of defining the actual calculation to be carried out.

Instead of programming the specific steps necessary at each variable point on the grid, it is proposed to allocate an integer code indicating the nature of each such point and later to deduce the required calculations from this numerical code. In this way the task of programming the finite-difference calculations is split into two distinct stages:

a) numerical coding of the geometry of the model domain,

b) provision of a general time-stepping subroutine which can perform the computation required at each grid-point, given the corresponding code.

The major advantage of this approach is that although stage a) must be carried out each time a new modelling problem is tackled, stage b) need be done only once, should the same finite-difference method be suitable for subsequent models. This eliminates a recurring source of programming errors.

Leendertse [1] made partial use of this concept and later Heaps [2] implemented it fully; however the method is cumbersome when applied to the particular finite-difference scheme used in the latter paper. Because both components were defined at all velocity-points, there were 19 distinct types of velocity point even though only two types of boundary were permitted. This made it difficult to code model geometry correctly, particularly in the absence of independent checking programs. An important part of the system described in this report is a plotting program which permits convenient graphical checks on the correctness of the numerical coding done at stage a). The model boundaries are plotted, using different styles of line (e.g. full, broken, dashed) to indicate boundaries according to type, thus permitting rapid detection of errors in coding of the model geometry.

To summarize the proposed method, the user is required to choose a Cartesian grid for the area being modelled and to provide a corresponding array of mean water depths. The position and nature of the various boundaries are then coded, as explained in detail in §5 and the coding is checked using pre-developed graphical programs. The numerical model is then programmed, all the necessary finite-difference calculations being carried out in a pre-developed time-stepping subroutine. The user must provide information, in the form of a subroutine, concerning any forces or time-dependent boundary conditions influencing the water body. A sample model is discussed in §10.

A model programmed as proposed here takes roughly 20% more computing time than an equivalent model in which the finite-difference equations are programmed directly for the particular grid in question, but the saving in programming time is substantial. Memory requirements are increased by the need to store arrays of integer codes : the various methods which can be used to minimize this additional storage are not considered here, as, in general, they are machine-dependent.

The programs described are written in ANSI FORTRAN 77.

2. Governing Equations

The programs discussed in this report apply to problems which can be described adequately by the partly linearized shallow-water equations

$$\eta_t = -(du)_x - (dv)_y \tag{1}$$

$$u_t = -g\eta_x + fv - F^{(u)} + G^{(u)} \tag{2}$$

$$v_t = -g\eta_y - fu - F^{(v)} + G^{(v)} \tag{3}$$

where

$\eta(x,y,t)$ = elevation of water surface above mean level
$u(x,y,t)$ = depth-averaged velocity in x-direction
$v(x,y,t)$ = depth-averaged velocity in y-direction
$d(x,y)$ = mean water depth
x,y = Cartesian coordinates in horizontal plane
f = Coriolis coefficient (assumed constant)
g = acceleration due to gravity
t = time

$F^{(u)}$ and $F^{(v)}$ represent friction terms. In one of the stepping subroutines supplied (VLCA01), provision is made for friction being linearly proportional to velocity, i.e.

$$F^{(u)} = ru \quad ; \quad F^{(v)} = rv \quad , \tag{4}$$

where r is a (linear) friction coefficient. An alternative subroutine (VQCA01) uses the more commonly assumed quadratic friction forms:

$$F^{(u)} = ku(u^2 + v^2)^{\frac{1}{2}}/d \quad ; \quad F^{(v)} = kv(u^2 + v^2)^{\frac{1}{2}}/d \tag{5}$$

where k is a non-dimensional (quadratic) friction coefficient. The stepping subroutines currently available use constant values of r or k throughout the model domain.

The terms $G^{(u)}$ and $G^{(v)}$ in equations (1) - (3) represent forcing terms, such as surface wind stress, equilibrium tide gradient, etc, which may vary with x,y,t. Appropriate values for these must be specified for the particular problem being considered, by means of a user-provided subroutine (§7).

3. The Finite-Difference Scheme.

The simple Richardson grid shown in Figure 1 was chosen as the basis for the finite-difference scheme on the grounds that it minimizes storage and permits particularly simple simulation of coastlines. At interior points of the grid, equations (1) to (3) are represented by the following finite-difference forms:

$$\frac{\eta'_{ij} - \eta_{ij}}{\Delta t} = - \frac{(d_{ij} + d_{i+1,j})u_{i+1,j} - (d_{i-1,j} + d_{ij})u_{ij}}{2.\Delta x} - \frac{(d_{ij} + d_{i,j+1})v_{i,j+1} - (d_{i,j-1} + d_{ij})v_{ij}}{2.\Delta y} \tag{6}$$

$$\frac{u'_{ij} - u_{ij}}{\Delta t} = - g \frac{\eta'_{ij} - \eta'_{i-1,j}}{\Delta x} + f\tilde{v}_{ij} - F^{(u)}_{ij} + G^{(u)}_{ij} \tag{7}$$

$$\frac{v'_{ij} - v_{ij}}{\Delta t} = - g \frac{\eta'_{ij} - \eta'_{i,j-1}}{\Delta y} - f\tilde{u}'_{ij} - F^{(v)}_{ij} + G^{(v)}_{ij} \tag{8}$$

where

Δt = time step

$\Delta x, \Delta y$ = grid interval sizes in in x,y directions respectively

d_{ij} = mean water depth at elevation point η_{ij}

$$\tilde{u}_{ij} = \frac{1}{4} [u_{i,j-1} + u_{i+1,j-1} + u_{ij} + u_{i+1,j}] \tag{9}$$

$$\tilde{v}_{ij} = \frac{1}{4} [v_{i-1,j} + v_{ij} + v_{i-1,j+1} + v_{i,j+1}] \tag{10}$$

Primes indicate variables updated during the current time step; unprimed variables are those evaluated at the previous step. The use of old (unprimed) values of v in the Coriolis term in (7) and new (primed) values of u in the corresponding term in (8) is necessary for stability [3]. Fortunately, it also eliminates the need to store any but the most recently updated values of each variable, provided that the equations are applied in the order given, that is, at each time step, all η_{ij} are updated, then all the u_{ij}, and finally all the v_{ij}. The same stability and storage conclusions apply if variables are evaluated in the order η,v,u, using old values of u in the v-equation and new values of v in the u-equation. To reduce possible bias, the stepping subroutines VLCA01 and VQCA01 evaluate the variables in the order η', u', v', on odd-numbered steps and η', v',u', on even-numbered steps.

Strictly speaking, equations (6)-(8) imply that u_{ij} and v_{ij} are evaluated $\Delta t/2$ later than η_{ij}, but normally they are regarded as pertaining to the same time level. The distinction is important only when calculating quantities which depend on phase differences between elevation and velocity, for example, energy flux, and then only when there are relatively few time steps per wave period.

4. Choice of Grid.

In finite-difference models, coastlines have to be represented approximately by line segments of a rectangular grid. Often the need to obtain a reasonable fit to an actual coastline is the principal factor determining the choice of mesh size for the grid. Figure 2 shows a typical situation, where the mesh size and orientation of the grid have been chosen to give a compromise between the following requirements:

(i) the closest boundaries of the model should fit the actual coastlines with reasonable accuracy

(ii) that one of the coordinate axes of the grid should be aligned with the expected principal direction of currents at the sea boundary

In theory, requirement (i) can be satisfied to any required degree simply by reducing the grid mesh size sufficiently. However, computer storage and time limitations often dictate the minimum practicable mesh size. Since the minimum time step permissible for computational stability with equations (6)-(8) is related to mesh size by [3]

$$\Delta t \leq \frac{\Delta x . \Delta y}{[g d_{max}(\Delta x^2 + \Delta y^2)]^{1/2}} \qquad *(11)$$

reducing the mesh size by a factor of 2, for instance, involves halving the time step as well as quadrupling the number of variables to be stored, thus increasing by a factor of 8 the computation time required to run the model for a fixed length of real time.

5. Numerical Coding of Grid Geometry.

5.1 Introduction

The working programs described in the report constitute a simplified version of an automated system being developed for converting information about model coastlines into specification of the required finite-difference formulas. An overall view of the final system is shown in Figure 3. When the whole set of programs is available, the normal starting point will be at Box A, Figure 3: a video display with graphics tablet or cursor will be used to overlay a suitable grid on the area being modelled and subsequently to select which line segments of the grid are to represent the various boundaries. A program represented by Box B will then convert this information into "primary" codes (integer arrays KE, KU, KV), which define the general nature of the finite-difference treatment required at all η, u and v points respectively on the model grid.

---

* N.B. See §5.2.6.

Since graphical entry of model geometry is likely to be hardware-dependent, work on steps A and B has been postponed and, for the present, the normal starting point is Box F. This means that the user enters the codes KE,KU,KV directly in numerical form, as detailed in §5.2, and then, proceeding to step G, uses a graphical program (GKCA01) to check for coding errors.

Some processing of the coded information is required before it can be used in a stepping subroutine. In order to set up the finite-difference equations correctly in even the simplest linear model, it is necessary to distinguish between similar types of boundary with different orientations. For instance, different finite-difference calculations are generally required at land boundaries on opposite sides of a model. For such reasons, the primary codes are next converted by a program (BICA01), represented by Box C, Figure 3, into more complicated codes, IE,IU,IV, here termed "intermediate codes", which express the necessary distinctions between different boundaries. The intermediate codes are defined in detail in Figure 9, in case some user wishes to improve or modify the stepping subroutines; otherwise, there is no need to concern oneself with details of the intermediate codes. The conversion program also computes the bounds of the model area to be scanned by the stepping subroutine at each time-step (see §8).

When this approach to model construction was first tried, specifying what are now the intermediate codes was the first step in the coding process (Box H). Even with the aid of a graphical checking program (Box I), this takes considerably greater time and effort than defining the model geometry by means of primary codes (Boxes F,G and C) and the latter entry method has superseded the former.

The intermediate codes are adequate to specify all the different calculations required in problems involving linear or quadratic friction but no advection terms, that is, models using subroutines VLCA01 (Box J) or VQCA01 (Box K). When the equations of motion (6)-(8) are amended to cope with non-linear effects, such as advection, one-sided (asymmetric)

definitions of various terms are necessary at boundaries. Furthermore, where two boundaries form a corner, special formulas are needed to cope with the overlapping asymmetric calculation regimes. Consequently, when at a later stage of this project, a subroutine (Box E) is developed to handle the time-stepping finite-difference calculations for the fully nonlinear shallow water equations, it will be necessary to convert the intermediate codes at Box D to an even more diversified set of "secondary codes", NE,NU and NV, capable of distinguishing all the various combinations of asymmetric boundary formulas required. Work is in progress on programs to perform steps D and E.

It is intended also to use the numerical codes to specify the areas for active computation in programs designed to analyse the output of numerical models. For example, production of a cotidal chart from the output of a tidal model involves harmonic analysis and contouring over sea areas only; the region of the grid over which to carry out the appropriate calculations could be derived automatically from the numerical codes.

5.2 Primary Codes KE, KU, KV

The nature of the variable points η_{ij}, u_{ij}, v_{ij} associated with the ij^{th} mesh of the model grid (Figure 1) is conveyed by respective integer primary code numbers KE(I,J), KU(I,J), KV(I,J). The arrays KE,KU,KV have the same dimensions as the arrays η_{ij}, u_{ij}, v_{ij} respectively.

5.2.1 Interior Points: KE, KU, KV = 0

An elevation point, η, in the interior of a model, not directly affected by boundary conditions, is governed by equation (6) and is given a primary code KE = 0. Similarly, interior u-points and v-points, where the finite-difference calculations required are (7) or (8) respectively, are given codes KU = 0 or KV = 0 (see Figure 4).

5.2.2 Closed Boundaries: KU or KV = 1

As explained in §4, coastlines are represented by strings of line segments on the model grid. Assuming that there is no volume transport through such boundaries, they can be represented by putting the appropriate velocity component, u or v, identically equal to zero at every u- or v-point lying on line segments representing the coastline. The time-stepping subroutines VLCA01 and VQCA01 are instructed to set the appropriate velocity component to zero by setting the corresponding primary codes to KV = 1, as for example in Figure 4.

5.2.3 Spits or Causeways: KU or KV = 2

A spit or causeway or a narrow barrier island can be represented by putting u = 0 or v = 0 as appropriate on one or more suitably chosen line segments. For later purposes, a distinction has to be drawn between these variable-points and those on closed boundaries representing coastlines. For this reason, the code KU = 2 or KV = 2 is used in these cases where a narrow barrier separates two bodies of water.

Figure 4 shows a sample model grid illustrating all the primary codes introduced so far. It may be noted that while the area being modelled can be covered effectively by a 6 x 5 grid, an extra column of u-points and an extra row of v-points must be included to represent the right-hand and upper boundaries respectively. Thus appropriate dimensions for the variables η_{ij}, u_{ij}, and v_{ij} in this case would be 6 x 5, 7 x 5, and 6 x 6 respectively. The arrays KE, KU, KV are of corresponding size.

Where the code KE, KU or KV = 0 appears in Figure 4, finite-difference calculations are never required since the corresponding variable point lies on land. At these points the codes should be set equal to zero nevertheless, as if they were interior points. In the primary to secondary

conversion program BICA01 (Box C, Figure 3) bounds are computed for the area to be actively scanned at each time-step by the stepping subroutines and variable-points on land or outside the boundaries of the model are subsequently ignored.

5.2.4 Sea Boundaries with Specified Elevation: KE = 1

Where the area of sea being modelled is contiguous with another water body, the boundary line between the two is termed a sea boundary or sometimes an open boundary. In general, the two water bodies will interact, or to be more specific in the present context, long waves may pass in all directions across the boundary. In this case, the model is ill-posed unless suitable boundary conditions can be provided. This may be possible from theoretical knowledge or on the basis of tide gauge readings in the case of surface elevation, or current meter measurements in the case of velocity.

Where a model has a sea boundary at which elevation can be specified as a function of time, the code for elevation-points which lie on this boundary is KE = 1. An example is shown in Figure 5. The user supplies a subroutine, here arbitrarily called UFORCE, which, when called just prior to the stepping subroutine at each step, places the updated values of boundary elevations in certain designated arrays (see §7),ready for use in the stepping subroutine.

It should be noted that since the stepping subroutines always require η, u, v arrays with dimensions M x N, (M+1) x N, M x (N+1) respectively, for the model in Figure 5 these arrays must be 7 x 5, 8 x 5 and 7 x 6, i.e. the u array has an extra, dummy column in this case. Once again, the arrays KE, KU, KV should have the same dimensions as η,u,v respectively.

The expressions used in equations (7) and (8) for the Coriolis terms are not suitable for velocity points lying on a sea boundary where elevation is specified, because the spatial averaging involves velocity

points outside the model boundary. It was decided to use one-sided approximations in all such cases, using the mean of the nearest two velocity components inside the boundary. For example, in the model shown in Figure 5, the stepping subroutine would represent the velocity in the Coriolis term for v_{73} (point A) by the expression

$$\tilde{u}_{73} = \frac{1}{2} [u_{72} + u_{73}]$$

rather than by equation (10). Boundary velocity points of this type should be allocated codes KU = 0 or KV = 0; the necessary program changes are arranged automatically by setting appropriate intermediate codes IU or IV from the primary codes KE through execution of subroutine BICA01.

5.2.5. Sea Boundary with Specified Velocity: KU or KV = 3

A model may have sea boundaries at which velocity normal to the boundary is specified as a function of time. The primary code for a u-point or v-point lying on such a boundary is KU = 3 or KV = 3 respectively. An example is shown in Figure 5. Again, the updated values of the boundary velocities required are placed in designated arrays by the user-written subroutine UFORCE (see §7) at each time step.

5.2.6 Sea Boundary with Radiation Condition: KU or KV = 4

Where there are good grounds for assuming that no waves enter the model area from an adjacent water body, it is appropriate to use a radiation condition on the sea boundary between the two. This permits waves reaching the sea boundary from the interior of the model to pass out of the model domain.

When choosing the model grid initially, radiating sea boundaries parallel to the x-axis of the model should be placed to run through v-points on the grid (as illustrated in Figure 5). Similarly, those parallel to the y-axis should run through u-points. It is assumed that the radiation problem can be treated one-dimensionally at each

velocity point on the sea boundary and thus that the surface elevation and normal velocity at the boundary are related by

$$\text{outward normal velocity} = (g/d)^{1/2} \times \text{elevation}$$

Since there are no elevation points actually on the boundary, the nearest interior elevation value is taken instead, so that the formulas used in the stepping subroutines for u-points on radiating boundaries facing in the positive or negative x-direction are respectively:

$$u_{ij} = (g/d_{i-1,j})^{1/2} \cdot \eta_{i-1,j}$$

or

$$u_{ij} = -(g/d_{ij})^{1/2} \cdot \eta_{ij}$$

Similarly, at radiating sea boundaries facing in the positive or negative y-directions, the formulas used are respectively:

$$v_{ij} = (g/d_{i,j-1})^{1/2} \cdot \eta_{i,j-1}$$

or

$$v_{ij} = -(g/d_{ij})^{1/2} \cdot \eta_{ij}$$

For example, in the model shown in Figure 5, at each time step, the new velocity values v'_{46}, v'_{56}, on the radiating boundary are found from the newly-updated elevations η'_{45}, η'_{55} by putting

$$v'_{46} = (g/d_{45})^{1/2} \cdot \eta'_{45} \; ; \quad v'_{56} = (g/d_{55})^{1/2} \cdot \eta'_{55}$$

This simple but effective radiation condition was introduced by Heaps [4]. In practice, transmission across the boundary is nearly complete for waves impinging normally on the boundary, but there is considerable unwanted reflection when the angle of incidence exceeds 45^0. Tests show

that use of the radiation condition can reduce the permissible time step by as much as 50%. Equation (11) should therefore be amended to:

$$\Delta t \leq \frac{\Delta x . \Delta y}{2[gd_{max}(\Delta x^2 + \Delta y^2)]^{1/2}} \qquad (12)$$

to ensure stability in models using the above type of radiating boundary. In practice it may be found possible to increase Δt to between 70 or 80% of the value given by (11) without causing instability.

5.2.7 Summary of Primary Codes

Elevation Points

KE = 1 specified elevation at sea boundary
= 0 elsewhere

Velocity Points

KU = 1	KV = 1	coastline, land boundary
= 2	= 2	spit or causeway
= 3	= 3	specified velocity at sea boundary
= 4	= 4	radiating sea boundary
= 0	= 0	elsewhere

6. Graphical Check of Numerical Coding

In order to check that model geometry has been coded correctly, a program (GKCA01) is provided which plots the model boundaries (Box G, Figure 3) as represented by the primary codes KE, KU and KV. The plotting scale can be set equal to that on which the model grid is laid out, so that by overlaying the plot on the chart the correctness of the numerical coding

can be readily checked. The plotting program uses standard CALCOMP graphical routines and is documented internally (see Appendix 1).

It is recommended that the coding be checked in this manner whenever the codes are changed for any reason. In fact, it is this facility for checking the coding of the model geometry which makes this system of automatic programming really practicable.

A similar program (GICA01) is available for plotting the boundaries using the intermediate codes IE, IU, IV as input, but is not listed in this report since it is very unlikely that future users will choose to enter the intermediate codes directly.

7. Provision of Boundary and Forcing Information

The user is expected to provide a subroutine UFORCE, which supplies appropriate values of the forcing stresses $G_{ij}^{(u)}$ and $G_{ij}^{(v)}$ at u and v points, and whatever values of η_{ij}, u_{ij} or v_{ij} are specified at sea boundaries (§§5.2.4, 5.2.5). Since in general these values will be time-dependent, e.g. wind stresses, this subroutine will normally be called at each time step, just prior to the stepping subroutines, as shown in Figure 7. An example of this type of subroutine, corresponding to the sample model discussed in §10, is shown in Figure 4.

The values provided by UFORCE must be stored in specific arrays in order to be available to the stepping subroutine. The forcing terms $G_{ij}^{(u)}$ and $G_{ij}^{(v)}$ are stored in arrays (GU and GV in App. 3, 4) which have the same dimensions as u_{ij} and v_{ij} respectively.

In transferring values of η_{ij}, u_{ij}, v_{ij} for sea boundaries from the user subroutine UFORCE to the stepping subroutines, some economy in storage has been sacrificed for the sake of convenience and generality.

Arrays BL, BR, BB and BT are defined for storing the required values at the left, right, bottom and top edges of the model grid, respectively. It helps if these arrays are visualized as lying parallel to the edges of the grid, as shown in Figure 6, which refers to the same model as Figure 5. Projecting straight outwards from each variable point on a sea boundary where a value has to be supplied shows the location in the associated storage array where UFORCE should place the required value. Thus, for the example in Figure 6, updated boundary values are stored as follows:

u'_{11}	$\leftrightarrow$ BL(1)	η'_{72}	$\leftrightarrow$ BR(2)
u'_{12}	$\leftrightarrow$ BL(2)	η'_{73}	$\leftrightarrow$ BR(3)
u'_{13}	$\leftrightarrow$ BL(3)	η'_{74}	$\leftrightarrow$ BR(4)
v'_{41}	$\leftrightarrow$ BB(4)	v'_{46}	$\leftrightarrow$ BT(4)
v'_{51}	$\leftrightarrow$ BB(5)	v'_{56}	$\leftrightarrow$ BT(5)

With this system, some locations in the boundary storage arrays are not used, but this is counterweighed by the convenience of having storage array subscripts identical to one of the grid variable subscripts. Updated boundary values cannot in general be placed directly into the grid variable arrays, because some values from the previous time step may be required in the finite-difference formulas.

8. Conversion of Codes and Calculation of Limits

After the primary codes KE, KU, KV, have been verified and stored on file, they are converted (at stage C, Figure 3) to intermediate codes IE, IU, IV, by a subroutine BICA01 provided (see Appendix 2). Subsequently, in the same program, the area of the grid which must be scanned at each time step by the stepping subroutine is deduced from the intermediate codes. It is assumed that the finite-difference calculations will be performed column by column through the grid. A lower limit JB and an upper limit JT is worked out for the range of grid meshes in each column for which computations are required in the stepping subroutine. This information

enables the stepping subroutine to ignore variable points lying on the grid but outside the model domain and so economizes in computing time.

9. Procedure for Programming a Model

Assuming that the primary codes have been entered, checked and converted, the output of the conversion program (Box C, Figure 3), consisting of files containing the intermediate codes IE, IU, IV and column limits JB, JT, is now available for use in the model. Other requirements are a user-written subroutine UFORCE for providing boundary values (see §7 and example, Appendix 4) and a file containing an array of mean water depths, D.

The flowchart in Figure 7 shows the essential steps common to most models and the following notes refer to correspondingly labelled boxes in that figure:

(A) Appropriate values of physical parameters, such as f,g,k are read in from file. Also entered here are program parameters to specify the length of run,to control the choice of starting conditions at step C, to set the intervals at which output is to take place (step G) and to specify which type of output is required. Information about the physical layout of the model is entered here by reading in files, prepared by the user, containing mean water depths and the primary codes.

B) Intermediate codes and column limits are computed from the primary codes using the conversion program BICA01 (§8) and stored on file for use in the event of restart.

C) On the first run, a model is normally started from a state of rest (all η, u, v = 0), since synoptic initial conditions are hardly ever known. In tidal simulation, for example, a model is often started from rest and run for a few cycles. If analysis of the output shows that starting transients are still significant, the simulation is resumed, (steps I, J) for several more cycles.

D) It is convenient in many cases to choose the time step so that some integer multiple of Δt is equal to a convenient time interval in the problem considered, e.g. one cycle in the case of a tidal model. Of course Δt must not exceed the limit dictated by stability requirements (see §§2, 5.2.6).

E) The updated values specified in §7 are computed in subroutine UFORCE, provided by the user, and stored in the arrays GU, GV, BL, BR, BB, BT for use in step F (see example, Appendix 4).

F) For this step, the user chooses between the two stepping subroutines VLCA01 and VQCA01, depending on whether the governing equations have linear or quadratic friction. The argument lists of these subroutines differ in only two terms (see Appendix 3) and consequently it is easy to replace one with the other.

G) The selection of variables to be output has to be left to the user, since the information required depends on the problem considered. One general recommendation that can be made is that all the variables should be saved at a few intermediate times during each run, in the same format as the final output (step H), so that the run can be resumed (steps I, J) if interrupted for any reason.

H) The final output should be adequate to define the complete state of the model, in order to permit resumption of the run if necessary, (steps I, J).

J) A dummy step, i.e. a call to UFORCE followed by a call to VQCA01 with Δt set equal to zero, is required to evaluate $\tilde{u}_{ij}$ and $\tilde{v}_{ij}$, when VQCA01 is used. VLCA01 does not use $\tilde{u}_{ij}$, $\tilde{v}_{ij}$, so not storing $\tilde{u}_{ij}$, $\tilde{v}_{ij}$, at step H facilitates replacement of VLCA01 with VQCA01 if required.

10. A Sample Model

A very simple tidal model involving the grid shown in Figure 5 will be treated here as an illustration. The resulting program is listed in Appendix 4.

The mean water depths d_{ij} in metres are shown in the following table:

J \ I	1	2	3	4	5	6	7
5	-	-	-	167	165	-	-
4	-	-	111	155	164	156	153
3	172	174	161	145	156	145	148
2	151	162	150	123	125	132	130
1	120	91	83	90	95	-	-

The forcing terms $G^{(u)}$ and $G^{(v)}$ in equations (2),(3), for a tidal model originate from gravitational effects and in reality are so small that their effects in this coastal model would be negligible. Simply to make their contribution to the computed motion detectable, these terms will be assumed here to be 10^4 times larger than their true physical values. Only the dominant semi-diurnal lunar tide, M_2, which has frequency $\omega = 1.4052.10^{-4}\ s^{-1}$ will be considered.

Supposing that the x-axis of the model points 30^0 north of east, the corresponding forcing terms in equations (2) and (3) are then

$$\begin{aligned} G(u) &= -3.76.10^{-4}\sin(\omega t + 4.78.10^{-7}x - 2.76.10^{-7}y) \\ G(v) &= 2.17.10^{-4}\sin(\omega t + 4.78.10^{-7}x - 2.76.10^{-7}y) \end{aligned} \qquad (13)$$

where the coordinate origin is at B in Figure 5.

The following conditions are given at the sea boundaries:-

$$
\begin{aligned}
u_{11} &= 0.21 \sin \omega t \text{ m/s} & \eta_{72} &= 0.88 \sin(\omega t - 0.135) \text{ m} \\
u_{12} &= 0.22 \sin \omega t \text{ m/s} & \eta_{73} &= 0.89 \sin(\omega t - 0.138) \text{ m} \\
u_{13} &= 0.23 \sin \omega t \text{ m/s} & \eta_{74} &= 0.90 \sin(\omega t - 0.140) \text{ m} \\
v_{41} &= 0.12 \sin (\omega t - 0.077) \text{ m/s} \\
v_{42} &= 0.13 \sin (\omega t - 0.096) \text{ m/s}
\end{aligned}
\tag{14}
$$

Other parameter values given are

$$
\begin{aligned}
f &= 1.2.10^{-4} \text{ s}^{-1} \\
g &= 9.81 \text{ m/s}^2 \\
\Delta x &= 6000 \text{ m} \\
\Delta y &= 5000 \text{ m}
\end{aligned}
$$

To minimize starting transients, each of the forcing terms (13) and boundary conditions (14) is replaced with zero in the computer program until its first zero crossing occurs. Thus the boundary conditions u_{11}, u_{12}, u_{13} in (4) apply from the start at t=0, but the term η_{72} is treated in effect as

$$
\begin{aligned}
\eta_{72} &= 0 && \text{for } t < 0.135/\omega \\
&= 0.89 \sin (\omega t - 0.135), && \text{for } t \geq 0.135/\omega
\end{aligned}
$$

It is assumed that at t=0, the system is at rest, that is, all η, u,v equal zero. The maximum permissible time step is 46.49 s according to (12). The number of time steps per tidal cycle was chosen to be 972, which gives an actual time step of 48.00187s.

The values computed for η, u, v at the end of steps 22 and 23 (by which point all the boundary driving terms are in effect) are included in Appendix 4, for checking purposes. After 10 cycles, most of the transient starting error is damped out and the solution is periodic within 1% error. The corresponding amplitudes and phases for all the variables, based on the 11th cycle, are listed at the end of Appendix 4. Each variable is assumed to be in the form A sin $(\omega t + \phi)$, where A is amplitude and ϕ is phase angle.

Acknowledgements

The writer is most grateful to Kathie Holtham for helpful suggestions and programming through much of this project, to Joe Mildenburger for acting as test pilot, to Coralie Wallace for preparing the diagrams and to Pat Perras for typing this report.

References

1. Leendertse, J.J. 1967. Aspects of a computational model for long-period water-wave propagation. Memo RM-5294-PR, RAND Corp., Santa Monica.

2. Sielecki, A. 1968. An energy-conserving difference scheme for the storm surge equations. Mon. Weather Rev., Vol. 96, pp. 150-156.

3. Heaps, N.S. 1969. A two-dimensional numerical sea model. Phil. Trans. Roy. Soc., London, Ser. A, Vol. 265, pp. 93-137.

4. Heaps, N.S. 1974. Development of a three-dimensional numerical model of the Irish Sea. Rapp. P.-v. Réun. Cons. Int. Explor. Mer, Vol. 167, pp. 147-162.

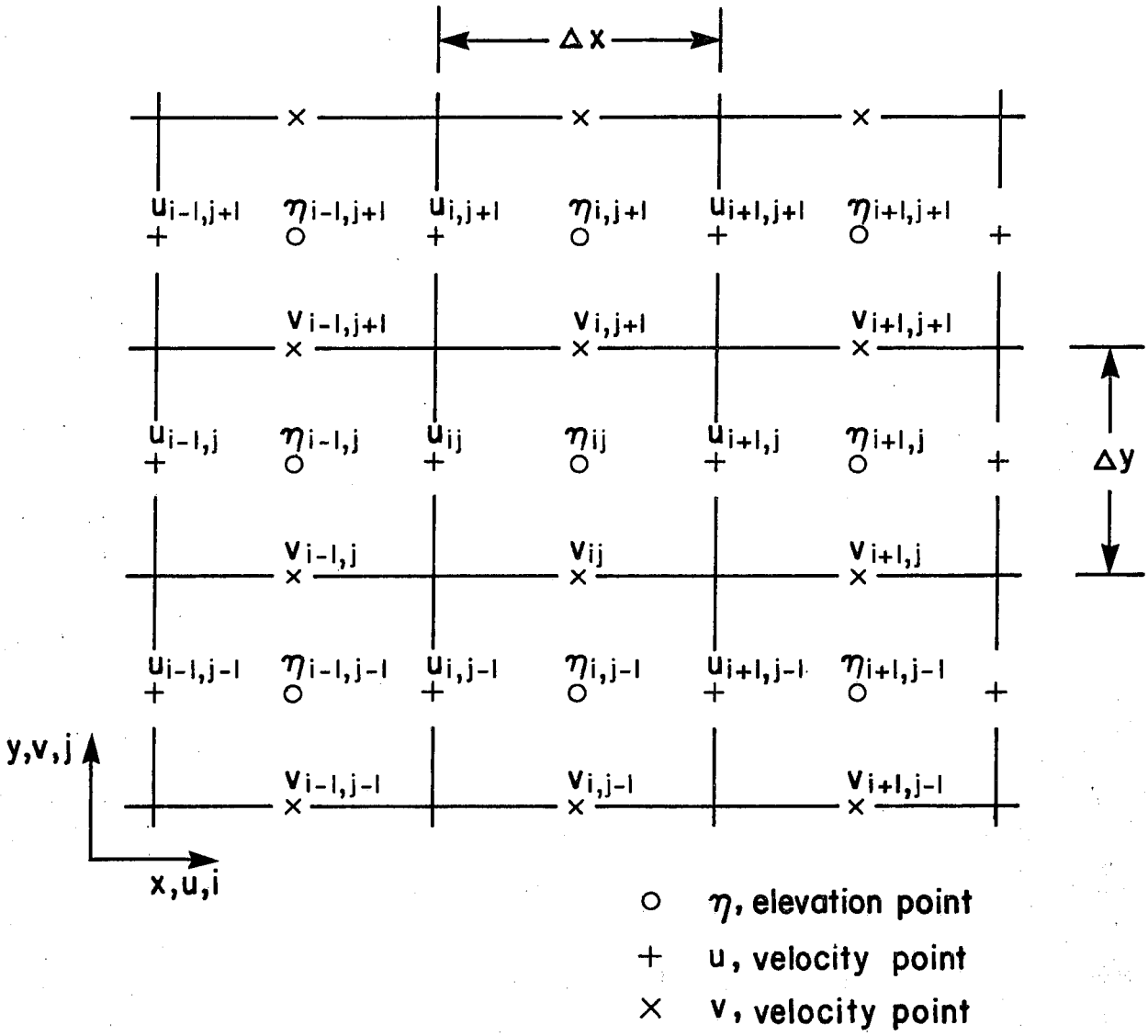


Fig. 1 Portion of Interior of Richardson Grid

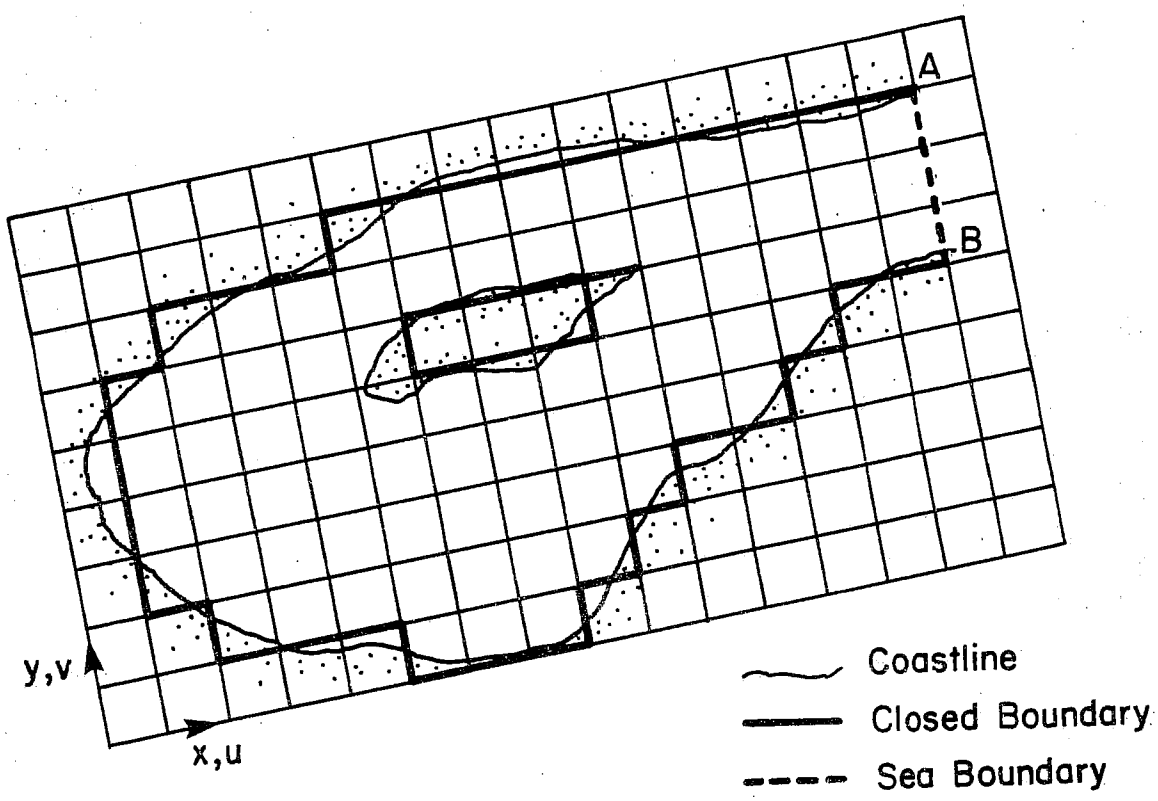


Fig. 2 Example of Basin with Grid

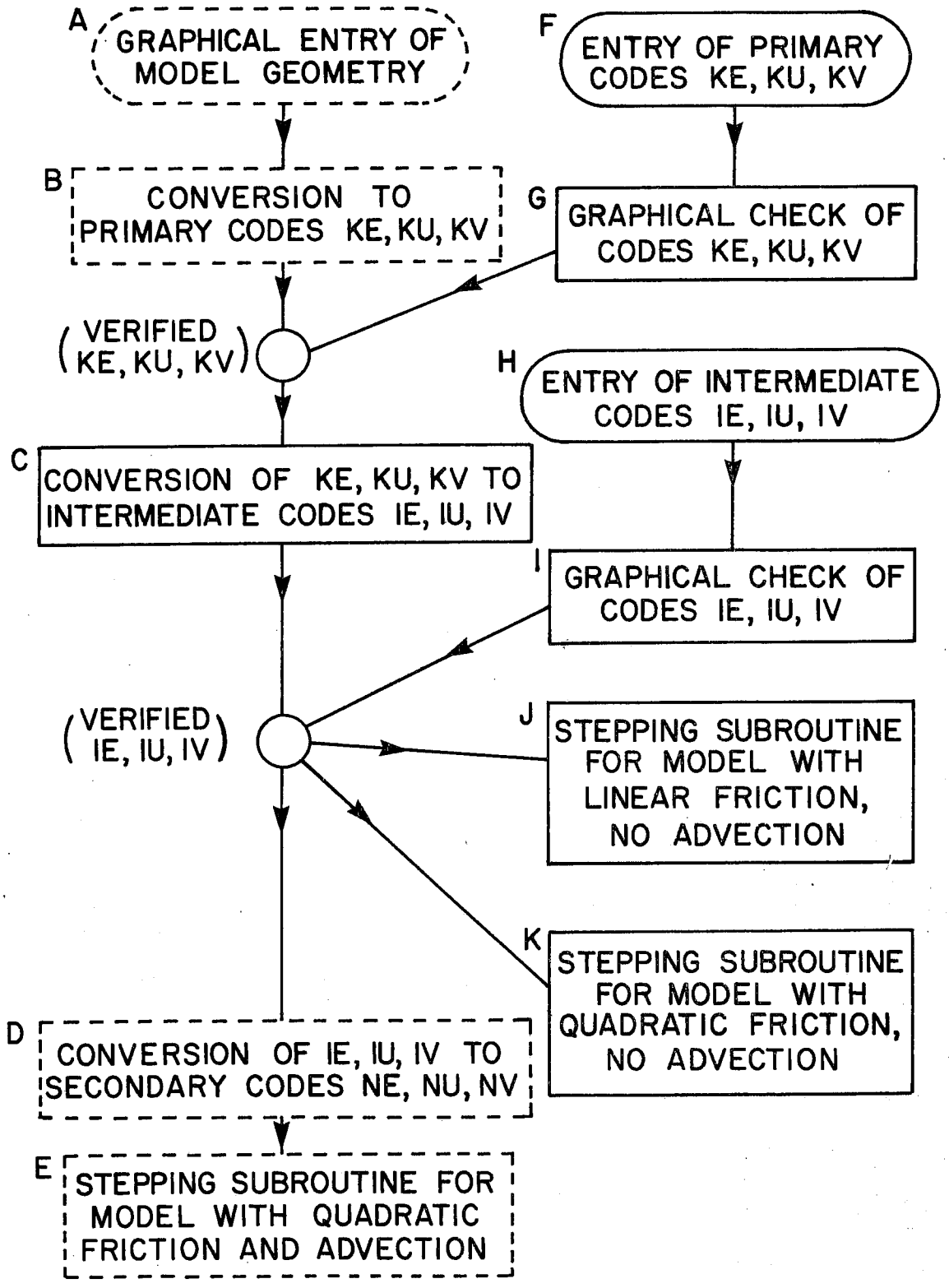
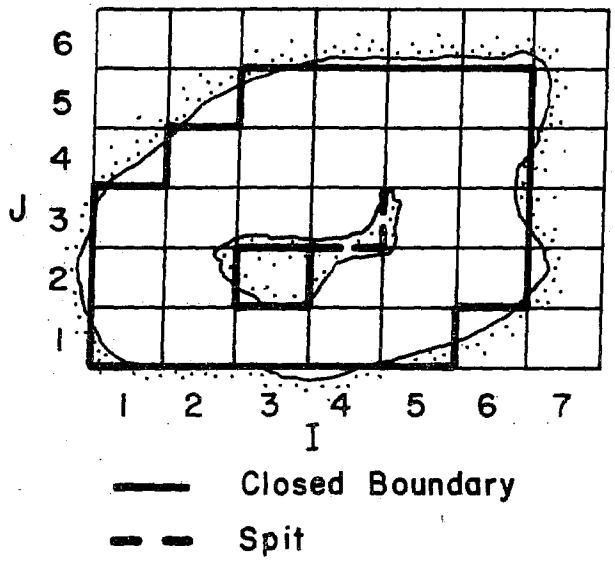


Fig. 3 Flowchart for Programming System



KE (for η-points)

J \ I	1	2	3	4	5	6
5	0	0	0	0	0	0
4	0	0	0	0	0	0
3	0	0	0	0	0	0
2	0	0	0	0	0	0
1	0	0	0	0	0	0

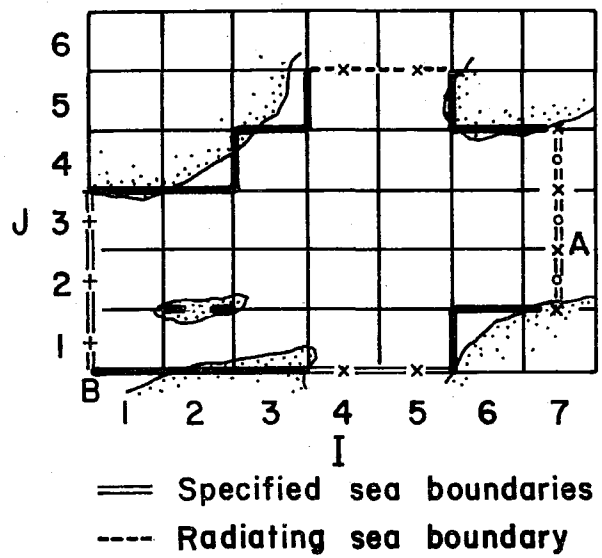
KU (for u-points)

J \ I	1	2	3	4	5	6	7
5	0	0	1	0	0	0	1
4	0	1	0	0	0	0	1
3	1	0	0	0	2	0	1
2	1	0	1	1	0	0	1
1	1	0	0	0	0	1	0

KV (for v-points)

J \ I	1	2	3	4	5	6
6	0	0	1	1	1	1
5	0	1	0	0	0	0
4	1	0	0	0	0	0
3	0	0	1	2	0	0
2	0	0	1	0	0	1
1	1	1	1	1	1	0

Fig.4 Sample Model Grid Showing Coding of Interior Points Closed Boundaries and Spits



KE (for η-points)

J \ I	1	2	3	4	5	6	7
5	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1
3	0	0	0	0	0	0	1
2	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0

KU (for u-points)

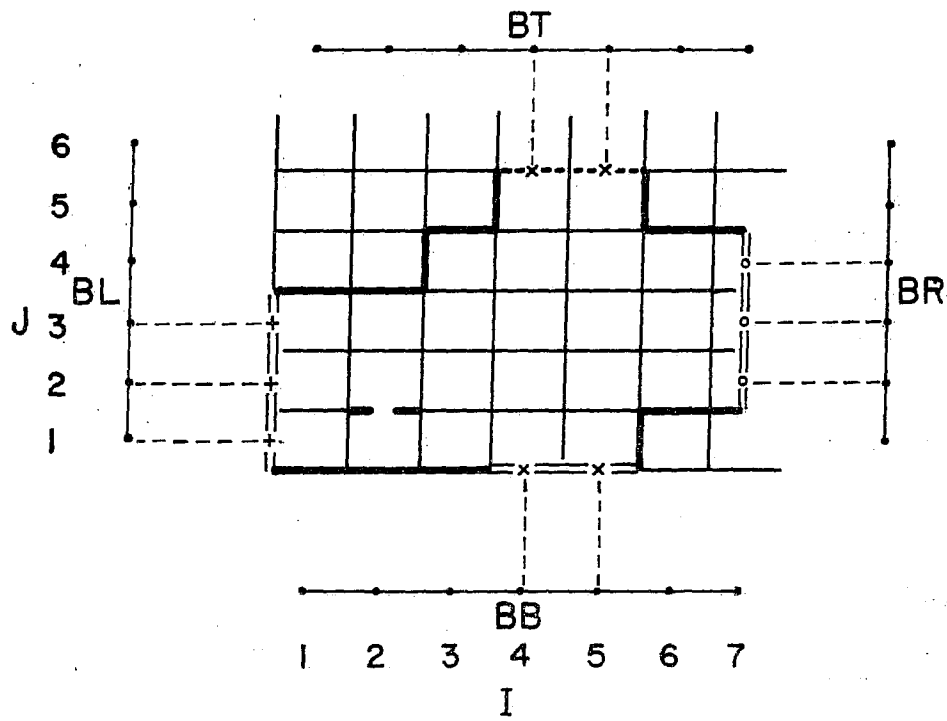
J \ I	1	2	3	4	5	6	7	8
5	0	0	0	1	0	1	0	0
4	0	0	1	0	0	0	0	0
3	3	0	0	0	0	0	0	0
2	3	0	0	0	0	0	0	0
1	3	0	0	0	0	1	0	0

KV (for v-points)

J \ I	1	2	3	4	5	6	7
6	0	0	0	4	4	0	0
5	0	0	1	0	0	1	1
4	1	1	0	0	0	0	0
3	0	0	0	0	0	0	0
2	0	2	0	0	0	1	1
1	1	1	1	3	3	0	0

Sample Model Grid Showing Coding of Sea Boundaries

Fig 5



Example of Storage Arrays for Sea Boundary Values

Fig. 6

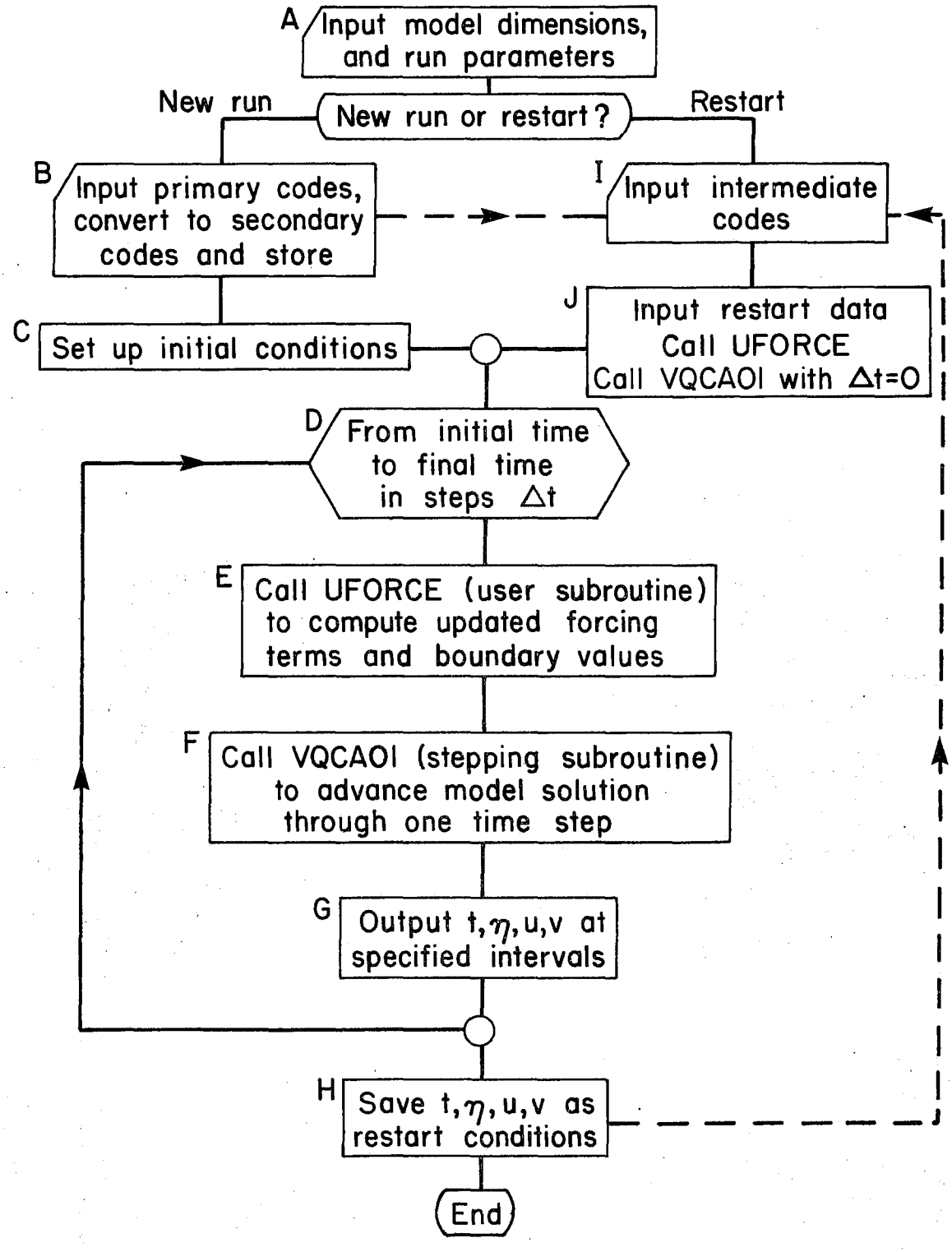
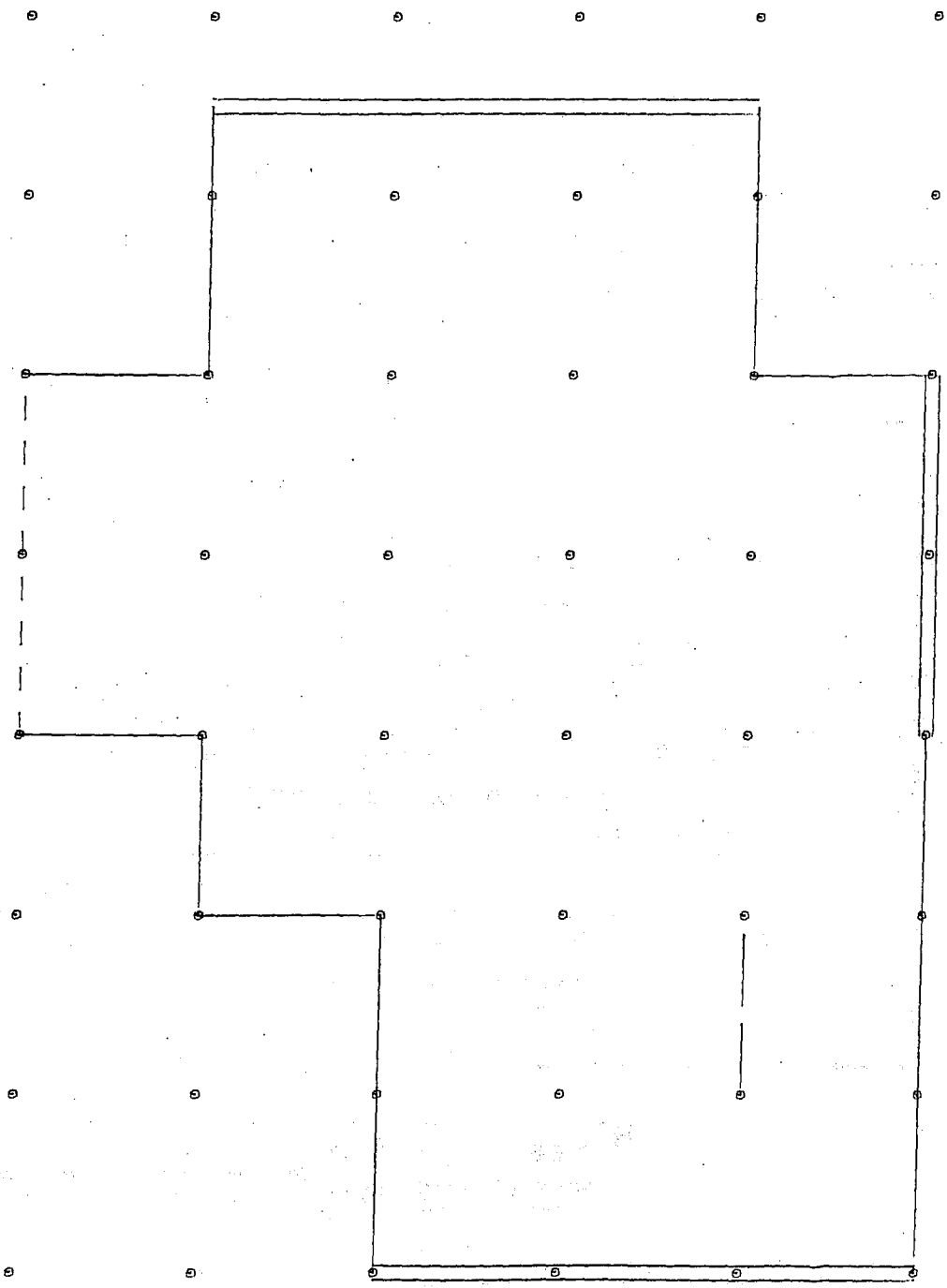
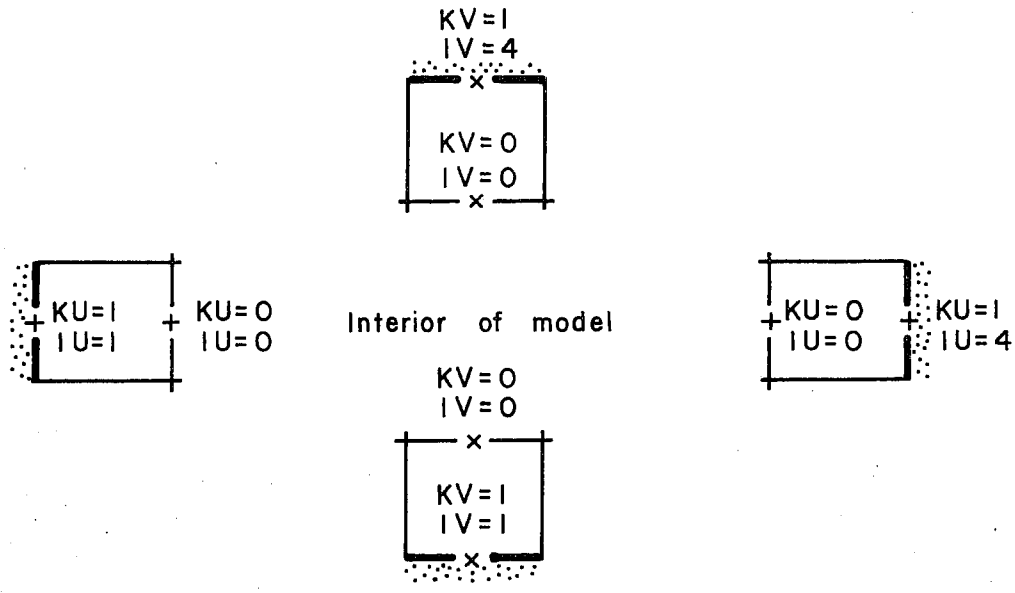


Fig. 7 Flowchart for Typical Model



Grid Produced from Primary Codes for Sample Model

Fig. 8

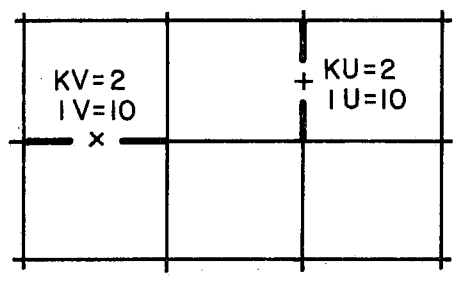


i) Interior points; closed boundaries

o η - point

+ U - point

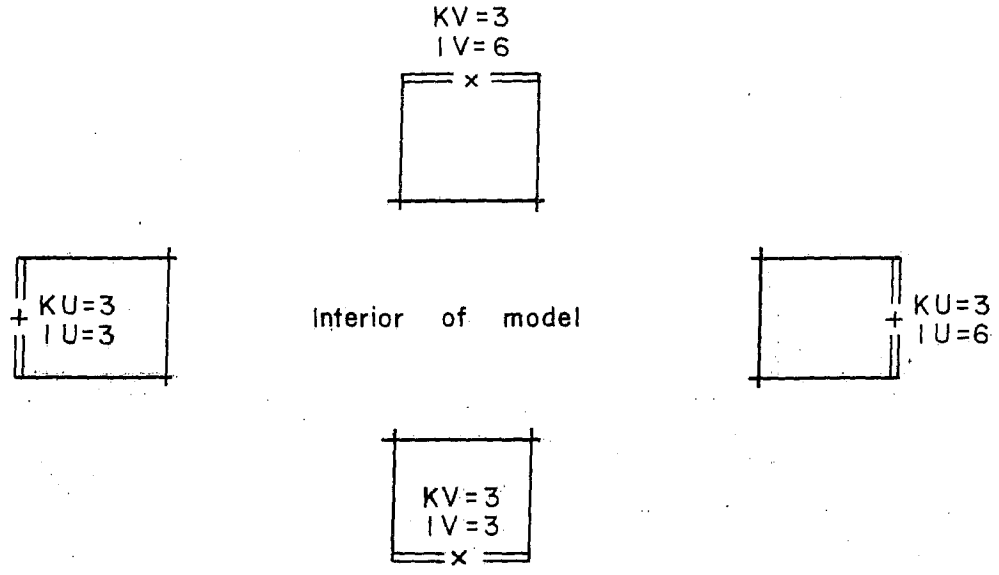
× V - point



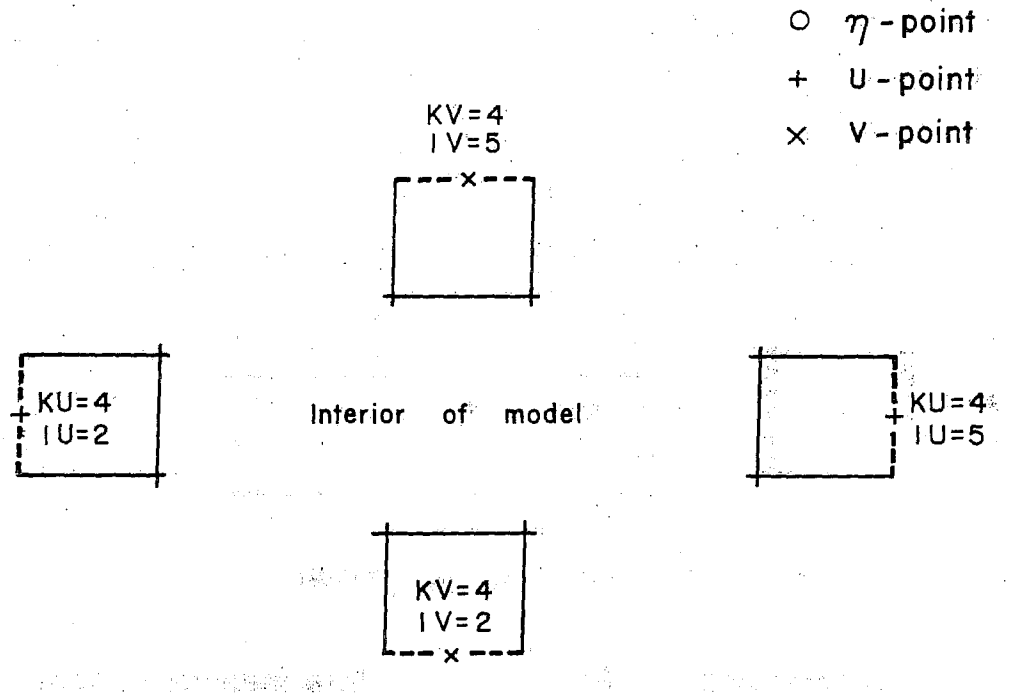
ii) Spits or causeways

Corresponding Primary and Intermediate Codes

Fig. 9a



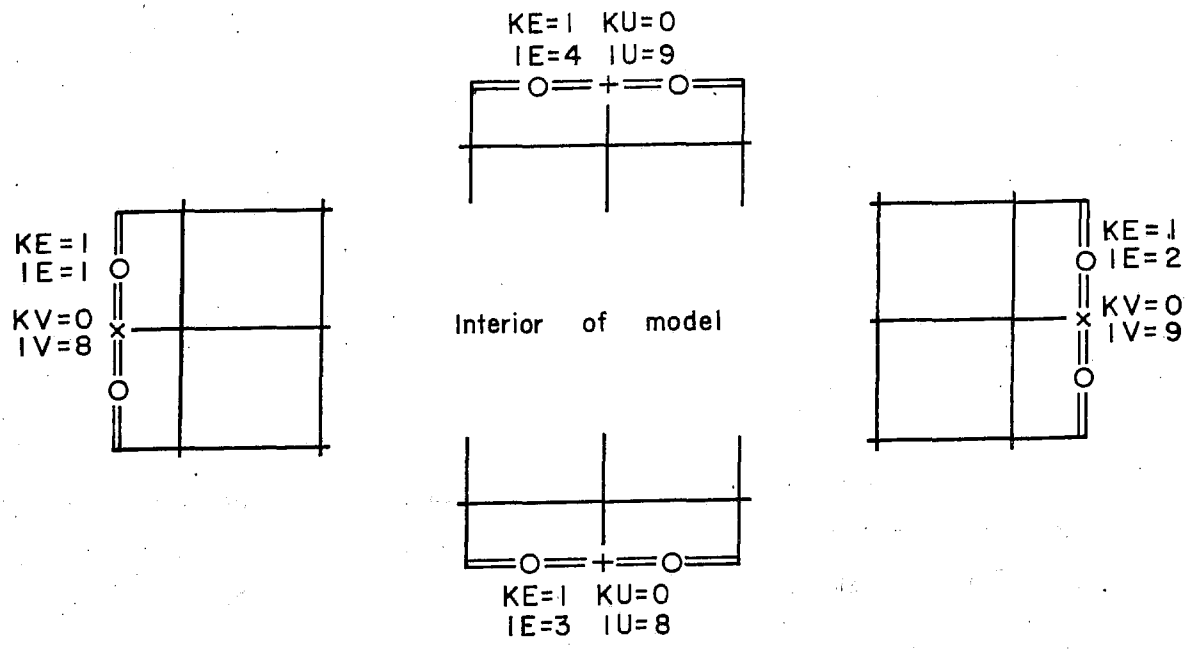
iii) Sea boundaries with specified velocities.



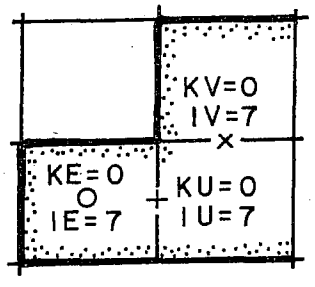
iv) Sea boundaries with radiation condition.

Corresponding Primary and Intermediate Codes

Fig. 9b



v) Sea boundaries with specified elevations.



○ η - point

\+ U - point

× V - point

vi) Variable points on land or outside model.

Corresponding Primary and Intermediate Codes

Fig. 9c

APPENDIX 1 Graphical Checking Program GKCA01

The input to this program consists principally of files containing the primary codes KE, KU, KV prepared by the user. The output is a diagram showing the boundaries of the model according to the following line codes:-

———	closed boundaries
— —	spit or causeway
═══	sea boundary with specified variable
– – – –	radiating sea boundary

Use of this program is recommended for checking the primary codes when they are first prepared and later whenever the codes are changed for any purpose. The program uses standard CALCOMP plotting subroutines.

A sample output plot, produced with this program from the primary codes given in Figure 5, is shown in Figure 8.

Listing of Program GKCA01

```
C   PROGRAM GKCA01 TO PLOT BOUNDARIES OF MODEL GRID
C   SPECIFIED BY PRIMARY CODES KE,KU,KV
C
C
C   REQUIRES SUBPROGRAMS  KEHV,KESPEC,IEU1,IEU2,IEV1,IEV2,
C   PLTHDB,PLTVDB,READ,XRANGE,YRANGE (ALL LISTED BELOW) AND
C   CALCOMP SUBROUTINES DASH,DASHLN,FRAME,PLOT,PLOTND,SYMBOL
C
C
C   I/O UNIT NUMBERS
C
C    IDUV      - UNIT NO.FOR FILE CONTAINING KE,KU,KV
C    ILINE     - UNIT NO. FOR LINEPRINTER OR EQUIVALENT
C    IREAD     - UNIT NO.FOR CARDREADER OR EQUIVALENT
C
C
C   INPUT VARIABLES
C
C    KE(I,J)   - PRIMARY CODE AT H(I,J)
C    KU(I,J)   - PRIMARY CODE AT U(I,J)
C    KV(I,J)   - PRIMARY CODE AT V(I,J)
C    M         - NO. OF GRID MESHES IN X-DIRECTION
C    N         - NO. OF GRID MESHES IN Y-DIRECTION
C     NOTE  DIMENSIONS IN COMMON STATEMENTS MUST BE ALTERED
C           IF M OR N EXCEEDS 50
C
C
C   PLOT PARAMETERS
C
C    STEPX     - PLOTTED LENGTH IN INCHES OF ONE GRID
C                INTERVAL IN X-DIRECTION
C    STEPY     - PLOTTED LENGTH IN INCHES OF ONE GRID
C                INTERVAL IN Y-DIRECTION
C    DST       - SPACING IN INCHES BETWEEN DOUBLE LINES ON
C                BOUNDARIES WHERE VARIABLES ARE SPECIFIED
C
C   GRAPHICAL CODES FOR BOUNDARY SEGMENTS ARE AS FOLLOWS
C    1) SOLID LINE REPRESENTS CLOSED BOUNDARY (KU OR KV = 1)
C    2) LONG DASHED LINE (TWO DASHES PER MESH) REPRESENTS A
C       SPIT OR CAUSEWAY (KU OR KV = 2)
C    3) SHORT DASHED LINE (FOUR DASHES PER MESH) REPRESENTS
C       A RADIATING SEA BOUNDARY ( KU OR KV = 4)
C    4) DOUBLE SOLID LINE REPRESENTS BOUNDARY WHERE ELEVATION
C       OR NORMAL VELOCITY IS SPECIFIED ( KE = 1, KU OR KV = 3)
C
      COMMON/CODES/KE(50,50),KU(50,50),KV(50,50)
      IDUV=3
      ILINE=6
```

```
      IREAD=5
      READ(IREAD,25)M,N,STEPX,STEPY,DST
25    FORMAT(I5/I5/F10.5/F10.5/F10.5)
      WRITE(ILINE,20) M,N,STEPX,STEPY,DST
20    FORMAT('1M =',I5,'  N = ',I5/' STEPX = ',F10.5,' STEPY = ',
     1F10.5,' DST = ',F10.5/)
      MP1=M+1
      NP1=N+1
C
C  READ IN PRIMARY CODES
C
      DO 23 J=N,1,-1
23    READ(IDUV,111) (KE(I,J),I=1,M)
      DO 21 J=N,1,-1
21    READ(IDUV,111) (KU(I,J),I=1,MP1)
      DO 22 J=NP1,1,-1
22    READ(IDUV,111) (KV(I,J),I=1,M)
111   FORMAT(40I2)
C
C  CHECK VALUES READ
C
      WRITE(ILINE,29)
29    FORMAT(//17X,'KE'/'  J')
      DO 28 J=N,1,-1
28    WRITE(ILINE,31) J,(KE(I,J),I=1,M)
      WRITE(ILINE,30)
30    FORMAT(//17X,'KU'/'  J')
      DO 32 J=N,1,-1
      WRITE(ILINE,31) J,(KU(I,J),I=1,MP1)
31    FORMAT(1X,I2,13X,50I2)
32    CONTINUE
      WRITE(ILINE,33)
33    FORMAT(//17X,'KV'/'  J')
      DO 34 J=NP1,1,-1
      WRITE(ILINE,31) J,(KV(I,J),I=1,M)
34    CONTINUE
      WRITE(ILINE,35)
35    FORMAT('1')
      CALL BDPLT(M,N,STEPX,STEPY,DST)
      STOP
      END


      SUBROUTINE BDPLT(M,N,STEPX,STEPY,DST)
C
C   THIS SUBROUTINE  CONTROLS THE PLOTTING
C
      COMMON /CODES/KE(50,50),KU(50,50),KV(50,50)
      MU=M+1
      NU=N
      MV=M
      NV=N+1
      DSHX1=STEPX*0.125
      DSHX21=STEPX*0.40
```

```
      DSHX22=STEPX*0.1
      DSHY1=STEPY*0.125
      DSHY21=STEPY*0.40
      DSHY22=STEPY*0.1
      CALL PLOT (0.0,0.0,3)
C
      D=DST/2.0
C
C   IN ORDER TO FACILITATE THE LOCATION OF BOUNDARIES WITHIN
C   THE GRID , THIS LOOP PLACES DOTS AT ALL GRID VERTICES.
      HT=0.02*STEPY
      DO 300 J=1,NV
      YPT=(J-1)*STEPY
      DO 300 I=1,MU
      XPT=(I-1)*STEPX
300   CALL SYMBOL(XPT,YPT,HT,1,0.,-1)
C   IF LARGER DOTS ARE DESIRED (ESPECIALLY FOR PAPER PLOT OUTPUT)
C   THEN HT SHOULD BE MADE A LARGER FRACTION OF STEPY
C
C   THIS LOOP PLOTS ALL BOUNDARIES ASSOCIATED WITH U GRID POINTS
C
      DO 510 I=1,MU
      SI=I
      X=(SI-1.0)*STEPX
      J=1
501   IDUA=KU(I,J)
      IF(IDUA.EQ.0) GO TO 500
      CALL YRANGE(I,J,MU,NU,STEPY,Y1,Y2)
      GO TO (200,240,260,230), IDUA
C
C   THE FOLLOWING STATEMENTS PLOT VERTICAL SOLID LINES
C
200   CALL PLOT(X,Y1,3)
      CALL PLOT(X,Y2,2)
      GO TO 500
C
C   THE FOLLOWING STATEMENTS DO LONG/SHORT DASHES
240   CALL DASHLN(DSHY21,DSHY22,DSHY21,DSHY22)
      CALL PLOT(X,Y1,3)
      CALL DASH(X,Y2)
      GO TO 500
C
C   THE FOLLOWING STATEMENTS PLOT SHORT DASHED LINES
C
230   CALL DASHLN(DSHY1,DSHY1,DSHY1,DSHY1)
      CALL PLOT(X,Y1,3)
      CALL DASH(X,Y2)
      GO TO 500
C
C   AND THESE STATEMENTS PLOT A DOUBLE SOLID LINE
C
260   CALL PLTVDB(X,Y1,Y2,0)
500   J=J+1
      IF(J.LE.NU) GO TO 501
510   CONTINUE
```

```
C
C   THIS LOOP PLOTS ALL BOUNDARIES ASSOCIATED WITH V GRID POINTS
C
      DO 910 J=1,NV
      SJ=J
      Y=(SJ-1.0)*STEPY
      I=1
901   IDVA=KV(I,J)
      IF(IDVA.EQ.0) GO TO 900
      CALL XRANGE(I,J,MV,NV,STEPX,X1,X2)
      GO TO (700,740,760,730),IDVA
C
C   THE FOLLOWING STATEMENTS PLOT HORIZONTAL SOLID LINES
C
700   CALL PLOT(X1,Y,3)
      CALL PLOT(X2,Y,2)
      GO TO 900
C
C   THE FOLLOWING STATEMENTS DO LONG/SHORT DASHES
C
740   CALL DASHLN(DSHX21,DSHX22,DSHX21,DSHX22)
      CALL PLOT(X1,Y,3)
      CALL DASH(X2,Y)
      GO TO 900
C
C   THE FOLLOWING STATEMENTS PLOT DASHED LINES
C
730   CALL DASHLN(DSHX1,DSHX1,DSHX1,DSHX1)
      CALL PLOT(X1,Y,3)
      CALL DASH(X2,Y)
      GO TO 900
C
C   AND THESE STATEMENTS PLOT A DOUBLE SOLID LINE
C
760   CALL PLTHDB(X1,X2,Y,D)
900   I=I+1
      IF(I.LE.MV) GO TO 901
  910 CONTINUE
C
C   AND THIS FINAL LOOP PLOTS ALL LINES REPRESENTING
C   BOUNDARIES WHERE ELEVATION IS SPECIFIED
C   THE PROCEDURE IS TO FIND A NONZERO KE VALUE, DETERMINE
C   WHETHER IT IS HORIZONTAL OR VERTICAL AND THEN SEE HOW
C   FAR THE BOUNDARY EXTENDS.  IT IS DRAWN, THE KE VALUES ARE
C   RESET TO ZERO AND THE PROCESS IS REPEATED.  THE SEARCH
C   FOR NON ZERO KE VALUES PROCEEDS BY ROW.
      DO 600 J=1,N
      DO 602 I=1,M
      IE=KE(I,J)
      IF(IE.EQ.0) GO TO 602
      CALL KESPEC(I,J,M,N,STEPX,STEPY,D)
602   CONTINUE
600   CONTINUE
      CALL FRAME
      CALL PLOTND
```

```
C
      STOP
      END



      SUBROUTINE YRANGE(I,J,MU,NU,STEPY,Y1,Y2)
      COMMON /CODES/KE(50,50),KU(50,50),KV(50,50)
C
C    THIS SUBROUTINE DETERMINES THE RANGE OF Y VALUES OVER WHICH
C    THE CURRENT LINE IS TO BE DRAWN
C    IT IS ASSUMED THAT KU(I,J) IS NON ZERO.
C
C    NOTE FOLLOWING ABBREVIATION
C         BWSE = BOUNDARY WITH SPECIFIED ELEVATION
C
      HSTEPY=0.5*STEPY
      J1=J
9     K=J+1
      IF(K.GT.NU) GO TO 10
      IF(KU(I,K).NE.KU(I,J)) GO TO 10
      J=K
      GO TO 9
10    J2=J
C    NOW CHECK IF THE UPPER OR LOWER GRID SEGMENTS SHOULD BE
C    SHORTENED BY HALF A GRID INTERVAL TO END AT HORIZONTAL BWSE.
      Y1=(J1-1)*STEPY
      Y2=J2*STEPY
      IF(J1.EQ.J2) GO TO 11
      IF(IEU1(I,J1,MU).NE.0) Y1=Y1+HSTEPY
      IF(IEU1(I,J2,MU).NE.0) Y2=Y2-HSTEPY
      RETURN
11    IF(IEU1(I,J1,MU).EQ.0) RETURN
C    SPECIAL CASE WHEN THE VERTICAL LINE IS ONLY 1 GRID
C    INTERVAL HIGH
      CALL IEU2(I,J1,MU,DY1,DY2)
      Y1=Y1+DY1*STEPY
      Y2=Y2-DY2*STEPY
      RETURN
      END



      INTEGER FUNCTION IEU1(I,J,MU)
C    THIS FUNCTION DETERMINES FOR A GIVEN COORDINATE POINT,
C    WHETHER OR NOT A HORIZONTAL ELEVATION BOUNDARY IS ADJACENT
      COMMON /CODES/KE(50,50),KU(50,50),KV(50,50)
      IE1=0
      IE2=0
      IF(I.NE.1) IE1=KE(I-1,J)
      IF(I.NE.MU) IE2=KE(I,J)
      IEU1=IE1+IE2
      RETURN
      END
```

```
      SUBROUTINE IEU2(I,J,MU,Y1,Y2)
C   THIS FUNCTION IS FOR THE SPECIAL CASE WHEN THE VERTICAL
C   LINE TO BE DRAWN IS ONLY ONE GRID SQUARE LONG AND
C   IS ADJACENT TO A HORIZONTAL BWSE.
      COMMON /CODES/KE(50,50),KU(50,50),KV(50,50)
      Y1=0.
      Y2=0.
C   THE APPROACH IS TO CHECK FOR NON ZERO VALUES IN THE
C   SURROUNDING(UP TO 4) KV VALUES.  THIS INDICATES IF BWSE IS
C   ABOVE OR BELOW VERTICAL LINE BEING DRAWN.
      IV=0
      IF(I.GT.1) IV=IV+KV(I-1,J)
      IF(I.LE.MU) IV=IV+KV(I,J)
C   IF IV.NE.0 THEN THERE IS A LAND BDY BELOW SO BWSE IS ABOVE
      IF(IV.EQ.0) GO TO 10
      Y2=0.5
      RETURN
10    IF(I.NE.1) IV=IV+KV(I-1,J+1)
      IF(I.LE.MU) IV=IV+KV(I,J+1)
      IF(IV.EQ.0) GO TO 11
C   THERE IS A LAND BDY ABOVE SO BWSE IS BELOW
      Y1=0.5
      RETURN
11    WRITE(ILINE,12) I,J
12    FORMAT('0CHECK THE CODES  THERE DOESNOT SEEM TO BE A LAND
     1BOUNDARY AROUND THESE U COORDINATES',2I5)
      STOP
      END


      SUBROUTINE XRANGE(I,J,MV,NV,STEPX,X1,X2)
      COMMON /CODES/KE(50,50),KU(50,50),KV(50,50)
C   THIS SUBROUTINE DETERMINES THE RANGE OF X VALUES OVER WHICH
C   THE CURRENT LINE IS TO BE DRAWN
      HSTEPX=0.5*STEPX
      I1=I
9     K=I+1
      IF(K.GT.MV) GO TO 10
      IF(KV(K,J).NE.KV(I,J)) GO TO 10
      I=K
      GO TO 9
10    I2=I
C   CHECK IF LEFT OR RIGHT GRID SEGMENTS SHOULD BE
C   SHORTENED BY HALF A GRID INTERVAL TO END AT VERTICAL BWSE.
      X1=(I1-1)*STEPX
      X2=I2*STEPX
      IF(I1.EQ.I2) GO TO 11
      IF(IEV1(I1,J,NV).NE.0) X1=X1+HSTEPX
      IF(IEV1(I2,J,NV).NE.0) X2=X2-HSTEPX
      RETURN
11    IF(IEV1(I1,J,NV).EQ.0) RETURN
```

```
      CALL IEV2(I1,J,NV,DX1,DX2)
      X1=X1+DX1*STEPX
      X2=X2-DX2*STEPX
      RETURN
      END


      INTEGER FUNCTION IEV1(I,J,NV)
C    THIS FUNCTION DETERMINES FOR A GIVEN COORDINATE POINT,
C    WHETHER OR NOT A VERTICAL ELEVATION BOUNDARY IS ADJACENT
      COMMON /CODES/KE(50,50),KU(50,50),KV(50,50)
      IE1=0
      IE2=0
      IF(J.NE.1) IE1=KE(I,J-1)
      IF(J.NE.NV) IE2=KE(I,J)
      IEV1=IE1+IE2
      RETURN
      END


      SUBROUTINE IEV2(I,J,NV,X1,X2)
C    THIS FUNCTION IS FOR THE SPECIAL CASE WHEN THE HORIZONTAL
C    LINE TO BE DRAWN IS ONLY ONE GRID INTERVAL LONG AND
C    IS ADJACENT TO A VERTICAL BWSE.
      COMMON /CODES/KE(50,50),KU(50,50),KV(50,50)
      X1=0.
      X2=0.
C    THE APPROACH IS TO CHECK FOR NON ZERO VALUES IN THE
C    SURROUNDING(UP TO 4) KU VALUES.  THIS INDICATES IF
C    VERTICAL BWSE IS TO LEFT OR RIGHT OF HORIZONTAL
C    LINE BEING DRAWN.
      IU=0
      IF(J.GT.1) IU=IU+KU(I,J-1)
      IF(J.LT.NV) IU=IU+KU(I,J)
C    IF IU.NE.0 THEN THERE IS A LAND BDY AT LEFT AND BWSE AT RIGHT
C    OF THE MODEL
      IF(IU.EQ.0) GO TO 10
      X2=0.5
      RETURN
10    IF(J.NE.1) IU=IU+KU(I+1,J-1)
      IF(J.LT.NV) IU=IU+KU(I+1,J)
      IF(IU.EQ.0) GO TO 11
C    THERE IS A LAND BOUNDARY AT RIGHT SO BWSE IS AT LEFT
      X1=0.5
      RETURN
11    WRITE(ILINE,12) I,J
12    FORMAT('0CHECK CODES  THERE DOES NOT SEEM TO BE A LAND
     1BOUNDARY AROUND THESE V COORDINATES',2I5)
      STOP
      END
```

```
      INTEGER FUNCTION KEHV(I,J,M,N)
C   THIS FUNCTION DETERMINES WHETHER OR NOT A BWSE IS
C   HORIZONTAL OR VERTICAL  KEHV=0 DENOTES HORIZONTAL,
C   KEHV=1 DENOTES VERTICAL, KEHV=2 DENOTES BOTH (CORNER
C   FORMED BY TWO BWSE'S).
      COMMON /CODES/KE(50,50),KU(50,50),KV(50,50)
      IE1=KV(I,J)+KV(I,J+1)
      IE2=KU(I,J)+KU(I+1,J)
      KEHV=0
      IF(IE2.GT.0) RETURN
      KEHV=1
      IF(IE1.GT.0) RETURN
C   THE THIRD POSSIBILITY IS A CORNER FORMED BY TWO BWSE'S
      KEHV=2
      IE3=0
      IF(J.GT.1) IE3=IE3+KE(I,J-1)
      IF(J.LT.N) IE3=IE3+KE(I,J+1)
      IF(I.GT.1) IE3=IE3+KE(I-1,J)
      IF(I.LT.M) IE3=IE3+KE(I+1,J)
      IF(IE3.EQ.2) RETURN
C   THE FINAL POSSIBILTY IS THAT ELEVATIONS ARE SPECIFIED
C   ALL THE WAY AROUND, IN WHICH CASE IE3 EQUALS 1.
      KEHV=3
      RETURN
      END


      SUBROUTINE PLTVDB(X,Y1,Y2,D)
C   THIS SUBROUTINE PLOTS A VERTICAL DOUBLE LINE.
      X1=X-D
      X2=X+D
      CALL PLOT(X1,Y1,3)
      CALL PLOT(X1,Y2,2)
      CALL PLOT(X2,Y2,3)
      CALL PLOT(X2,Y1,2)
      RETURN
      END


      SUBROUTINE PLTHDB(X1,X2,Y,D)
C   THIS SUBROUTINE PLOTS A A HORIZONTAL DOUBLE LINE
      Y1=Y-D
      Y2=Y+D
      CALL PLOT(X1,Y1,3)
      CALL PLOT(X2,Y1,2)
      CALL PLOT(X2,Y2,3)
      CALL PLOT(X1,Y2,2)
      RETURN
      END
```

```
      SUBROUTINE KESPEC(I,J,M,N,STEPX,STEPY,D)
C   THIS SUBROUTINE ASSUMES KE(I,J) IS NON ZERO AND IS THE
C   AT THE END OF AN ELEVATION BOUNDARY.  THIS BOUNDARY IS PLOTTED
C   AS FAR AS POSSIBLE, INCLUDING AROUND CORNERS.
      COMMON /CODES/KE(50,50),KU(50,50),KV(50,50)
      HSTEPX=0.5*STEPX
      HSTEPY=0.5*STEPY
      I0=I
      J0=J
      ISTOP=0
      GO TO (11,10,12,13), KEHV(I,J,M,N)+1
13    WRITE(ILINE,131) I,J
131   FORMAT('0KEHV CODE=3 INITIALLY FOR COORDINATES',2I5)
      CALL PLOTND
      STOP
C
C   THE BOUNDARY IS VERTICAL
C   ENTRY HERE IMPLIES THE BOUNDARY EXTENDS UPWARDS AND
C   AND IS ADJACENT TO V LAND BOUNDARY BELOW
10    Y1=(J-1)*STEPY
      X=(I-0.5)*STEPX
      JSTEP=1
      JLIMIT=N
C   ENTRY HERE IMPLIES A HORIZONTAL BWSE HAS ALREADY
C   BEEN PLOTTED.
100    L1=J+JSTEP
      DO 101 L=L1,JLIMIT,JSTEP
      IF(KE(I,L).NE.KE(I,J)) GO TO 102
101   CONTINUE
      J2=JLIMIT
      GO TO 103
102   J2=L-JSTEP
C   CHECK IF END POINT IS ADJACENT TO ANOTHER BWSE
C   THEN DRAW DOUBLE LINE
C   DOUBLE LINE
103    IEND=KEHV(I,J2,M,N)
      Y2=(J2+0.5*(JSTEP-1))*STEPY
      IF(IEND.GE.2) Y2=Y2-HSTEPY*JSTEP
      CALL PLTVDB(X,Y1,Y2,D)
C   RESET KE VALUES
      DO 104 L=J,J2,JSTEP
104   KE(I,L)=0
      IF(IEND.EQ.2) GO TO 106
      IF(ISTOP.EQ.1) GO TO 121
      KE(I0,J0)=0
      RETURN
C   PLOT HAS ENCOUNTERED A HORIZONTAL BWSE, THEREFORE CONTINUE.
C   KEEP KE(I0,J0)=1 IN EVENT OF RETURN TO STARTING POINT.
106   KE(I0,J0)=1
      KE(I,J2)=1
      X1=X
      Y=Y2
      J=J2
C   IE1=1 IMPLIES LINE GOES LEFT, IE2=1 LINE GOES RIGHT.
```

```
      IE1=0
      IE2=0
      IF(I.GT.1) IE1=KE(I-1,J2)
      IF(I.LT.M) IE2=KE(I+1,J2)
      IF(IE2.EQ.1) GO TO 105
      ISTEP=-1
      ILIMIT=1
      GO TO 110
105   ISTEP=1
      ILIMIT=M
      GO TO 110
C
C   THE BOUNDARY IS HORIZONTAL
C   ENTRY HERE IMPLIES BOUNDARY EXTENDS TO RIGHT AND
C   IS ADJACENT TO U LAND BOUNDARY
11    ISTEP=1
      ILIMIT=M
      Y=(J-0.5)*STEPY
      X1=(I-1)*STEPX
C   ENTRY HERE IMPLIES A VERTICAL BWSE HAS ALREADY
C   BEEN PLOTTED.
110   L1=I+ISTEP
      DO 111 L=L1,ILIMIT,ISTEP
      IF(KE(L,J).NE.KE(I,J)) GO TO 112
111   CONTINUE
      I2=ILIMIT
      GO TO 113
112   I2=L-ISTEP
C   CHECK IF THIS END POINT IS ADJACENT TO ANOTHER BWSE
C   ANOTHER ELEVATION SPECIFIED BOUNDARY
113   IEND=KEHV(I2,J,M,N)
      X2=(I2+0.5*(ISTEP-1))*STEPX
      IF(IEND.GE.2) X2=X2-HSTEPX*ISTEP
      CALL PLTHDB(X1,X2,Y,D)
C   RESET KE VALUES
      DO 114 L=I,I2,ISTEP
114   KE(L,J)=0
      IF(IEND.EQ.2) GO TO 116
      IF(ISTOP.EQ.1) GO TO 121
      KE(IO,JO)=0
      RETURN
C   PLOT HAS ENCOUNTERED A VERTICAL BWSE, THEREFORE CONTINUE.
C   KEEP KE(IO,JO)=1 IN EVENT OF RETURN TO STARTING POINT.
116   KE(IO,JO)=1
      KE(I2,J)=1
      I=I2
      Y1=Y
      X=X2
C   IE1=1 IMPLIES VERTICAL BWSE GOES DOWNWARD
C   IE2=1 IMPLIES VERTICAL BWSE GOES UPWARD
      IE1=0
      IE2=0
      IF(J.GT.1) IE1=KE(I,J-1)
      IF(J.LT.N) IE2=KE(I,J+1)
      IF(IE2.EQ.1) GO TO 115
```

```
      JSTEP=-1
      JLIMIT=1
      GO TO 100
115   JSTEP=1
      JLIMIT=N
      GO TO 100
C
C   PLOT IS AT CORNER FORMED BY TWO BWSE'S. CONTINUE
C   HORIZONTALLY AS FAR AS POSSIBLE. DUE TO MANNER OF
C   SEARCH, FIRST BOUNDARY MUST EXTEND TO THE RIGHT.
12    X1=(I-0.5)*STEPX
      Y=(J-0.5)*STEPY
      ISTEP=1
      ILIMIT=M
      ISTOP=1
      GO TO 110
C   RETURN TO PLOT VERTICAL LINE, IF NOT ALREADY PLOTTED.
C   DUE TO MANNER OF SEARCH, IT MUST EXTEND UPWARD
121   I=I0
      J=J0
      IF(KE(I,J+1).EQ.0) GO TO 122
      ISTOP=0
      Y1=(J-0.5)*STEPY
      X=(I-0.5)*STEPX
      JSTEP=1
      JLIMIT=N
      GO TO 100
122   KE(I0,J0)=0
      RETURN
      END
```

Appendix 2

Code Conversion Subroutine BICA01

```
      SUBROUTINE BICA01(IE,IU,IV,JB,JT,KE,KU,KV,M,MP1,N,NP1)
C
C        SUBROUTINE TO CONVERT PRIMARY GRID CODES KE,KU,KV
C        TO INTERMEDIATE CODES IE,IU,IV AND FIND LIMITS JB
C        AND JT FOR EACH COLUMN
C
C   IE(I,J)    - INTERMEDIATE CODE AT H(I,J)
C   IU(I,J)    - INTERMEDIATE CODE AT U(I,J)
C   IV(I,J)    - INTERMEDIATE CODE AT V(I,J)
C   JB(I)      - LOWEST J VALUE PROCESSED IN COLUMN I
C   JT(I)      - HIGHEST J VALUE PROCESSED IN COLUMN I
C   KE(I,J)    - PRIMARY CODE AT H(I,J)
C   KU(I,J)    - PRIMARY CODE AT U(I,J)
C   KV(I,J)    - PRIMARY CODE AT V(I,J)
C   M          - NUMBER OF GRID MESHES IN X-DIRECTION
C   MP1        - M+1
C   N          - NUMBER OF GRID MESHES IN Y DIRECTION
C   NP1        - N+1
C
      DIMENSION JB(MP1),JT(MP1)
      DIMENSION KU(MP1,N),IU(MP1,N)
      DIMENSION KV(M,NP1),IV(M,NP1)
      DIMENSION KE(M,N),IE(M,N)
      INTEGER EDGES
C
      DO 400 I=1,M
      DO 400 J=1,N
      IU(I,J)=0
      IV(I,J)=0
      IE(I,J)=0
400   CONTINUE
      DO 401 I=1,M
      IV(I,NP1)=0
401   CONTINUE
      DO 402 J=1,N
      IU(MP1,J)=0
402   CONTINUE
      DO 403 I=1,MP1
      JB(I)=0
      JT(I)=0
403   CONTINUE
C
C   CHANGING KU CODES TO IU CODES
C
      DO 2 J=1,N
      EDGES=0
      JP1=J+1
      JM1=J-1
```

```
      DO 2 I=1,MP1
      KUGO=KU(I,J)+1
      GO TO (5,10,20,30,40) KUGO
C   LAND
5     IF(EDGES.EQ.0.AND.IU(I,J).EQ.0) IU(I,J)=7
      GO TO 100
C   LAND BOUNDARIES
10    EDGES=EDGES+1
      IF(EDGES-2)11,12,100
11    IU(I,J)=1
      GO TO 100
12    IU(I,J)=4
      EDGES=0
      GO TO 100
C   SPITS
20    IU(I,J)=10
      GO TO 100
C   VELOCITY GIVEN
30    EDGES=EDGES+1
      IF(EDGES-2)31,32,100
31    IU(I,J)=3
      GO TO 100
32    IU(I,J)=6
      EDGES=0
      GO TO 100
C   OPEN RADIATING
40    EDGES=EDGES+1
      IF(EDGES-2)41,42,100
41    IU(I,J)=2
      GO TO 100
42    IU(I,J)=5
      EDGES=0
100   IF(I.EQ.MP1) GO TO 2
      IP1=I+1
      IM1=I-1
      IF(KE(I,J).EQ.0) GO TO 53
      EDGES=EDGES+1
      IF(EDGES-2) 51,52,2
C     LEFT HAND BOUNDARY(WESTERN)
51    IF((KE(I,JM1).EQ.1.OR.KE(I,JP1).EQ.1).AND.KE(IP1,J).EQ.0) THEN
         IE(I,J)=1
         IF(KE(I,JM1).EQ.1) IV(I,J)=8
         GO TO 2
      ENDIF
C     LOWER LEFT HAND CORNER(SOUTH WEST)
      IF(KE(I,JP1).EQ.1.AND.KE(I,JM1).EQ.0.AND.KE(IP1,J).EQ.1) THEN
         IE(I,J)=9
         IU(IP1,J)=8
         IV(I,JP1)=8
         GO TO 2
      ENDIF
C     UPPER LEFT HAND CORNER(NORTH WEST)
      IF(KE(I,JP1).EQ.0.AND.KE(I,JM1).EQ.1.AND.KE(IP1,J).EQ.1) THEN
          IE(I,J)=5
          IU(IP1,J)=9
```

```
          IV(I,J)=8
          GO TO 2
      ENDIF
      EDGES=EDGES-1
      GO TO 2
C     RIGHT HAND BOUNDARY(EASTERN)
52    IF((KE(I,JP1).EQ.1.OR.KE(I,JM1).EQ.1).AND.KE(IM1,J).EQ.0) THEN
         IE(I,J)=2
         IF(KE(I,JM1).EQ.1) IV(I,J)=9
         EDGES=0
         GO TO 2
      ENDIF
C     UPPER RIGHT HAND CORNER(NORTH EAST)
      IF(KE(IP1,J).EQ.0.AND.KE(IM1,J).EQ.1.AND.KE(I,JP1).EQ.0
     1.AND.KE(I,JM1).EQ.1) THEN
         IE(I,J)=6
         IU(I,J)=9
         IV(I,J)=9
         EDGES=0
         GO TO 2
      ENDIF
C     LOWER RIGHT HAND CORNER(SOUTH EAST)
      IF(KE(I,JP1).EQ.1.AND.KE(I,JM1).EQ.0.AND.KE(IM1,J).EQ.1
     1.AND.KE(IP1,J).EQ.0) THEN
         IE(I,J)=8
         IU(I,J)=8
         IV(I,JP1)=9
         EDGES=0
         GO TO 2
      ENDIF
      EDGES=EDGES-1
      GO TO 2
C     LAND  - OUTSIDE MODEL OR ISLAND
53    IF(IU(I,J).EQ.7.OR.IU(I,J).EQ.4) IE(I,J)=7
2     CONTINUE
C
C   CHANGING KV CODES TO IV CODES AND SETTING COLUMN LIMITS
C
      DO 3 I=1,M
      EDGES=0
      IP1=I+1
      IM1=I-1
      ILOWER=0
      DO 3 J=1,NP1
      KVGO=KV(I,J)+1
      GO TO (105,110,120,130,140) KVGO
C   LAND
105   IF(EDGES.EQ.0.AND.IV(I,J).EQ.0) IV(I,J)=7
      GO TO 200
C   LAND BOUNDARIES
110   EDGES=EDGES+1
      IF(EDGES-2)111,112,200
111   IV(I,J)=1
      IF(ILOWER.EQ.0) THEN
         JB(I)=J
```

```
            ILOWER=1
         ENDIF
         GO TO 200
112      IV(I,J)=4
         EDGES=0
         JT(I)=J-1
         GO TO 200
C      SPITS
120      IV(I,J)=10
         GO TO 200
C      VELOCITY GIVEN
130      EDGES=EDGES+1
         IF(EDGES-2)131,132,200
131      IV(I,J)=3
         IF(ILOWER.EQ.0) THEN
            JB(I)=J
            ILOWER=1
         ENDIF
         GO TO 200
132      IV(I,J)=6
         JT(I)=J-1
         EDGES=0
         GO TO 200
C      OPEN RADIATING
140      EDGES=EDGES+1
         IF(EDGES-2)141,142,200
141      IV(I,J)=2
         IF(ILOWER.EQ.0) THEN
            JB(I)=J
            ILOWER=1
         ENDIF
         GO TO 200
142      IV(I,J)=5
         JT(I)=J-1
         EDGES=0
200      IF(J.EQ.NP1) GO TO 3
         JP1=J+1
         JM1=J-1
         IEGO=IE(I,J)+1
         GO TO (150,3,3,150,150,6,6,154,4,4) IEGO
150      IF(KE(I,J).EQ.0) GO TO 153
         EDGES=EDGES+1
         IF(EDGES-2) 151,152,3
C        LOWER BOUNDARY(SOUTHERN)
151      IF((KE(IP1,J).EQ.1.OR.KE(IM1,J).EQ.1).AND.KE(I,JP1).EQ.0) THEN
            IE(I,J)=3
            IF(KE(IM1,J).EQ.1) IU(I,J)=8
            IF(ILOWER.EQ.0) THEN
               JB(I)=J
               ILOWER=1
            ENDIF
            GO TO 3
         ENDIF
         EDGES=EDGES-1
         GO TO 3
```

```
C       UPPER BOUNDARY(NORTHERN)
152     IF((KE(IP1,J).EQ.1.OR.KE(IM1,J).EQ.1).AND.KE(I,JM1).EQ.0) THEN
           IE(I,J)=4
           IF(KE(IM1,J).EQ.1) IU(I,J)=9
           EDGES=0
           JT(I)=J
           GO TO 3
        ENDIF
        EDGES=EDGES-1
        GO TO 3
C       LAND  - OUTSIDE MODEL OR ISLAND
153     IF(IV(I,J).EQ.4.OR.IV(I,J).EQ.7.OR.IU(IP1,J).EQ.7.OR.IU(IP1,J).
       1EQ.1) IE(I,J)=7
154     IF(IU(I,J).NE.7) THEN
           IF(ILOWER.EQ.0) THEN
              JB(I)=J
              ILOWER=1
           ELSE
              JT(I)=J
           ENDIF
        ENDIF
        GO TO 3
C   LOWER CORNER BOUNDARY
4       EDGES=EDGES+1
        IF(ILOWER.EQ.0) THEN
           JB(I)=J
           ILOWER=1
        ENDIF
        GO TO 3
C    UPPER CORNER BOUNDARY
6       EDGES=0
        JT(I)=J
3       CONTINUE
C
        DO 600 J=1,N
        IF(IU(MP1,J).NE.7) THEN
           JB(MP1)=J
           GO TO 601
        ENDIF
600     CONTINUE
601     DO 602 J=N,JB(MP1)+1,-1
        IF(IU(MP1,J).NE.7) THEN
           JT(MP1)=J
           GO TO 603
        ENDIF
602     CONTINUE
603     CONTINUE
        RETURN
        END
```

Appendix 3

Stepping Subroutines VLCA01 and VQCA01

```
      SUBROUTINE VLCA01(BB,BL,BR,BT,D,DT,DX,DY,F,G,GU,GV,H,IE,IU,IV,
     *JB,JT,M,MP1,N,NP1,NS,R,T,U,V)
C
C        VLCA01 IS A STEPPING SUBROUTINE FOR SOLVING LINEARIZED
C        SHALLOW WATER EQUATIONS WITH LINEAR FRICTION FOR A GRID
C        SIZE MXN. AN ALTERNATING HUV,HVU SCHEME IS USED.
C        THE IU,IV,IE GRID IDENTIFICATION SYSTEM IS USED
C
C   BB(I)       - BOTTOM BOUNDARY VALUE IN COLUMN I
C   BL(J)       - LEFT HAND BOUNDARY VALUE IN ROW J
C   BR(J)       - RIGHT HAND BOUNDARY VALUE IN ROW J
C   BT(I)       - TOP BOUNDARY VALUE IN COLUMN I
C   D(I,J)      - WATER DEPTH AT H(I,J)                    (M)
C   DT          - TIME STEP                                (S)
C   DX          - GRID INTERVAL IN X-DIRECTION             (M)
C   DY          - GRID INTERVAL IN Y-DIRECTION             (M)
C   F           - CORIOLIS COEFFICIENT                     (S**-1)
C   G           - ACCELERATION DUE TO GRAVITY              (M/S**2)
C   GU(I,J)     - FORCING TERM AT U(I,J)                   (M/S**2)
C   GV(I,J)     - FORCING TERM AT V(I,J)                   (M/S**2)
C   H(I,J)      - SURFACE ELEVATION,ETA                    (M)
C   IE(I,J)     - GRID CODE AT H(I,J)
C   IU(I,J)     - GRID CODE AT U(I,J)
C   IV(I,J)     - GRID CODE AT V(I,J)
C   JB(I)       - LOWEST J VALUE PROCESSED IN COLUMN I
C   JT(I)       - HIGHEST J VALUE PROCESSED IN COLUMN I
C   M           - NUMBER OF GRID MESHES IN X-DIRECTION
C   MP1         - M+1
C   N           - NUMBER OF GRID MESHES IN Y DIRECTION
C   NP1         - N+1
C   NS          - NUMBER OF TIME STEP
C   R           - LINEAR FRICTION COEFFICIENT              (S**-1)
C   T           - TIME                                     (S)
C   U(I,J)      - VELOCITY IN X-DIRECTION                  (M/S)
C   V(I,J)      - VELOCITY IN Y-DIRECTION                  (M/S)
C
      DIMENSION H(M,N),U(MP1,N),V(M,NP1),IU(MP1,N),IV(M,NP1),IE(M,N)
      DIMENSION D(M,N),JB(MP1),JT(MP1),FAC(6)
      DIMENSION GU(MP1,N),GV(M,NP1),BL(N),BR(N),BB(M),BT(M)
C
      FAC(1)=DT/(2.*DX)
      FAC(2)=DT/(2.*DY)
      FAC(3)=G*DT/DX
      FAC(4)=G*DT/DY
      FAC(5)=F*DT
      FAC(6)=R*DT
```

```
C
C     CALCULATE NEW SURFACE ELEVATIONS
C
      DO 9 I=1,M
      JBN=JB(I)
      JTN=JT(I)
      IF(JTN.EQ.0) GO TO 9
      IP1=I+1
      IM1=I-1
      DO 9 J=JBN,JTN
      JP1=J+1
      JM1=J-1
      IEGO=IE(I,J)+1
      GO TO (20,1,2,3,4,4,4,9,3,3) IEGO
20    RHS=0.
      IUGO=IU(I,J)+1
      GO TO (27,21,26,26,21,21,21,21,21,21,21) IUGO
26    RHS=RHS-U(I,J)*2.*D(I,J)
      GO TO 21
27    RHS=RHS-U(I,J)*(D(IM1,J)+D(I,J))
21    IUGO=IU(IP1,J)+1
      GO TO (29,22,22,22,22,28,28,22,22,22,22) IUGO
28    RHS=RHS+U(IP1,J)*2.*D(I,J)
      GO TO 22
29    RHS=RHS+U(IP1,J)*(D(I,J)+D(IP1,J))
22    RHS=RHS*FAC(1)
      RH=0.
      IVGO=IV(I,JP1)+1
      GO TO (31,23,23,23,23,30,30,23,23,23,23) IVGO
30    RH=RH+V(I,JP1)*2.*D(I,J)
      GO TO 23
31    RH=RH+V(I,JP1)*(D(I,JP1)+D(I,J))
23    IVGO=IV(I,J)+1
      GO TO (33,24,32,32,24,24,24,24,24,24,24) IVGO
32    RH=RH-V(I,J)*2.*D(I,J)
      GO TO 24
33    RH=RH-V(I,J)*(D(I,J)+D(I,JM1))
24    RH=RH*FAC(2)
      H(I,J)=H(I,J)-RHS-RH
      GO TO 9
1     H(I,J)=BL(J)
      GO TO 9
2     H(I,J)=BR(J)
      GO TO 9
3     H(I,J)=BB(I)
      GO TO 9
4     H(I,J)=BT(I)
9     CONTINUE
C
C      AN ALTERNATING UV,VU ORDER OF CALCULATION
C
      MOD=NS-2*(NS/2)
      IF(MOD.EQ.0) GO TO 10
```

```
12      CONTINUE
C
C        CALCULATE U COMPONENT VELOCITIES
C
        DO 100 I=1,MP1
        JBN=JB(I)
        JTN=JT(I)
        IF(JTN.EQ.0) GO TO 100
        IP1=I+1
        IM1=I-1
        DO 100 J=JBN,JTN
        JP1=J+1
        JM1=J-1
        IUGO=IU(I,J)+1
        GO TO (120,110,140,150,110,170,160,100,120,120,110) IUGO
110     U(I,J)=0.
        GO TO 100
120     IF(IUGO-9) 121,122,123
121     VVHAT=(V(I,J)+V(I,JP1)+V(IM1,J)+V(IM1,JP1))/4.
        GO TO 124
122     VVHAT=(V(I,JP1)+V(IM1,JP1))/2.
        GO TO 124
123     VVHAT=(V(I,J)+V(IM1,J))/2.
124     U(I,J)=U(I,J)-FAC(3)*(H(I,J)-H(IM1,J))+FAC(5)*VVHAT-FAC(6)*U(I,J)
        GO TO 100
140     U(I,J)=-SQRT(G/D(I,J))*H(I,J)
        GO TO 100
150     U(I,J)=BL(J)
        GO TO 100
160     U(I,J)=BR(J)
        GO TO 100
170     U(I,J)=SQRT(G/D(IM1,J))*H(IM1,J)
100     CONTINUE
C
        IF(MOD.EQ.0) GO TO 11
10      CONTINUE
C
C        CALCULATE V COMPONENT VELOCITIES
C
        DO 200 I=1,M
        JBN=JB(I)
        JTN=JT(I)+1
        IF(JT(I).EQ.0) GO TO 200
        IP1=I+1
        IM1=I-1
        DO 200 J=JBN,JTN
        IVGO=IV(I,J)+1
        JP1=J+1
        JM1=J-1
        GO TO (220,210,240,250,210,270,260,200,220,220,210) IVGO
210     V(I,J)=0.
        GO TO 200
220     IF(IVGO-9) 221,222,223
```

```
221   UUHAT=(U(I,J)+U(IP1,J)+U(I,JM1)+U(IP1,JM1))/4.
      GO TO 224
222   UUHAT=(U(IP1,J)+U(IP1,JM1))/2.
      GO TO 224
223   UUHAT=(U(I,J)+U(I,JM1))/2.
224   V(I,J)=V(I,J)-FAC(4)*(H(I,J)-H(I,JM1))-FAC(5)*UUHAT-FAC(6)*V(I,J)
      GO TO 200
240   V(I,J)=-SQRT(G/D(I,J))*H(I,J)
      GO TO 200
250   V(I,J)=BB(I)
      GO TO 200
260   V(I,J)=BT(I)
      GO TO 200
270   V(I,J)=SQRT(G/D(I,JM1))*H(I,JM1)
200   CONTINUE
C
      IF (MOD.EQ.0) GO TO 12
11    CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE VOCAD1(BB,BL,BR,BT,D,DT,DX,DY,F,G,GU,GV,H,IE,IU,IV,
     *JB,JT,M,MP1,N,NP1,NS,T,U,UHAT,V,VHAT,XK)
C
C        VOCAD1 IS A STEPPING SUBROUTINE FOR SOLVING LINEARIZED
C        SHALLOW WATER EQUATIONS WITH QUADRATIC FRICTION FOR A
C        GRID SIZE MXN. AN ALTERNATING HUV,HVU SCHEME IS USED.
C        THE IU,IV,IE GRID IDENTIFICATION SYSTEM IS USED
C
C   BB(I)      - BOTTOM BOUNDARY VALUE IN COLUMN I
C   BL(J)      - LEFT HAND BOUNDARY VALUE IN ROW J
C   BR(J)      - RIGHT HAND BOUNDARY VALUE IN ROW J
C   BT(I)      - TOP BOUNDARY VALUE IN COLUMN I
C   D(I,J)     - WATER DEPTH AT H(I,J)                    (M)
C   DT         - TIME STEP                                (S)
C   DX         - GRID INTERVAL IN X-DIRECTION             (M)
C   DY         - GRID INTERVAL IN Y-DIRECTION             (M)
C   F          - CORIOLIS COEFFICIENT                     (S**-1)
C   G          - ACCELERATION DUE TO GRAVITY              (M/S**2)
C   GU(I,J)    - FORCING TERM AT U(I,J)                   (M/S**2)
C   GV(I,J)    - FORCING TERM AT V(I,J)                   (M/S**2)
```

```
C     H(I,J)      - SURFACE ELEVATION,ETA                          (M)
C     IE(I,J)     - GRID CODE AT H(I,J)
C     IU(I,J)     - GRID CODE AT U(I,J)
C     IV(I,J)     - GRID CODE AT V(I,J)
C     JB(I)       - LOWEST J VALUE PROCESSED IN COLUMN I
C     JT(I)       - HIGHEST J VALUE PROCESSED IN COLUMN I
C     M           - NUMBER OF GRID MESHES IN X-DIRECTION
C     MP1         - M+1
C     N           - NUMBER OF GRID MESHES IN Y DIRECTION
C     NP1         - N+1
C     NS          - NUMBER OF TIME STEP
C     T           - TIME                                           (S)
C     U(I,J)      - VELOCITY IN X-DIRECTION                        (M/S)
C     UHAT(I,J)   - SPACE-AVERAGED U-VELOCITY AT V(I,J)            (M/S)
C     V(I,J)      - VELOCITY IN Y-DIRECTION                        (M/S)
C     VHAT(I,J)   - SPACE-AVERAGED V-VELOCITY AT U(I,J)            (M/S)
C     XK          - QUADRATIC FRICTION COEFFICIENT,K
C
      DIMENSION H(M,N),U(MP1,N),V(M,NP1),IU(MP1,N),IV(M,NP1),IE(M,N)
      DIMENSION D(M,N),JB(MP1),JT(MP1),UHAT(MP1,N),VHAT(M,NP1),FAC(6)
      DIMENSION GU(MP1,N),GV(M,NP1),BL(N),BR(N),BB(M),BT(M)
C
      FAC(1)=DT/(2.*DX)
      FAC(2)=DT/(2.*DY)
      FAC(3)=G*DT/DX
      FAC(4)=G*DT/DY
      FAC(5)=F*DT
      FAC(6)=2.*XK*DT
C
C     CALCULATE NEW SURFACE ELEVATIONS
C
      DO 9 I=1,M
      JBN=JB(I)
      JTN=JT(I)
      IF(JTN.EQ.0) GO TO 9
      IP1=I+1
      IM1=I-1
      DO 9 J=JBN,JTN
      JP1=J+1
      JM1=J-1
      IEGO=IE(I,J)+1
      GO TO (20,1,2,3,4,4,4,9,3,3) IEGO
20    RHS=0.
      IUGO=IU(I,J)+1
      GO TO (27,21,26,26,21,21,21,21,21,21,21) IUGO
26    RHS=RHS-U(I,J)*2.*D(I,J)
      GO TO 21
27    RHS=RHS-U(I,J)*(D(IM1,J)+D(I,J))
21    IUGO=IU(IP1,J)+1
      GO TO (29,22,22,22,22,28,28,22,22,22,22) IUGO
28    RHS=RHS+U(IP1,J)*2.*D(I,J)
      GO TO 22
29    RHS=RHS+U(IP1,J)*(D(I,J)+D(IP1,J))
```

```
22      RHS=RHS*FAC(1)
        RH=0.
        IVGO=IV(I,JP1)+1
        GO TO (31,23,23,23,23,30,30,23,23,23,23) IVGO
30      RH=RH+V(I,JP1)*2.*D(I,J)
        GO TO 23
31      RH=RH+V(I,JP1)*(D(I,JP1)+D(I,J))
23      IVGO=IV(I,J)+1
        GO TO (33,24,32,32,24,24,24,24,24,24,24) IVGO
32      RH=RH-V(I,J)*2.*D(I,J)
        GO TO 24
33      RH=RH-V(I,J)*(D(I,J)+D(I,JM1))
24      RH=RH*FAC(2)
        H(I,J)=H(I,J)-RHS-RH
        GO TO 9
1       H(I,J)=BL(J)
        GO TO 9
2       H(I,J)=BR(J)
        GO TO 9
3       H(I,J)=BB(I)
        GO TO 9
4       H(I,J)=BT(I)
9       CONTINUE
C
C        AN ALTERNATING UV,VU ORDER OF CALCULATION
C
        MOD=NS-2*(NS/2)
        IF(MOD.EQ.0) GO TO 10
12      CONTINUE
C
C        CALCULATE U COMPONENT VELOCITIES
C
        DO 100 I=1,MP1
        JBN=JB(I)
        JTN=JT(I)
        IF(JTN.EQ.0) GO TO 100
        IP1=I+1
        IM1=I-1
        DO 100 J=JBN,JTN
        JP1=J+1
        JM1=J-1
        IUGO=IU(I,J)+1
        GO TO (120,110,140,150,110,170,160,100,120,120,110) IUGO
110     U(I,J)=0.
        GO TO 100
120     DENOM=1.0+FAC(6)*SQRT(U(I,J)**2+VHAT(I,J)**2)/(D(I,J)+D(IM1,J))
        ELEV=FAC(3)*(H(I,J)-H(IM1,J))
        IF(MOD.NE.0) THEN
           U(I,J)=(U(I,J)+FAC(5)*VHAT(I,J)-ELEV+GU(I,J)*DT)/DENOM
        ELSE
           IF(IUGO-9) 121,122,123
121        VVHAT=(V(I,J)+V(I,JP1)+V(IM1,J)+V(IM1,JP1))/4.
           GO TO 124
```

```
122       VVHAT=(V(I,JP1)+V(IM1,JP1))/2.
          GO TO 124
123       VVHAT=(V(I,J)+V(IM1,J))/2.
124       U(I,J)=(U(I,J)+FAC(5)*VVHAT-ELEV+GU(I,J)*DT)/DENOM
          VHAT(I,J)=VVHAT
       ENDIF
       GO TO 100
140    U(I,J)=-SQRT(G/D(I,J))*H(I,J)
       GO TO 100
150    U(I,J)=BL(J)
       GO TO 100
160    U(I,J)=BR(J)
       GO TO 100
170    U(I,J)=SQRT(G/D(IM1,J))*H(IM1,J)
100    CONTINUE
C
       IF(MOD.EQ.0) GO TO 11
10     CONTINUE
C
C       CALCULATE V COMPONENT VELOCITIES
C
       DO 200 I=1,M
       JBN=JB(I)
       JTN=JT(I)+1
       IF(JT(I).EQ.0) GO TO 200
       IP1=I+1
       IM1=I-1
       DO 200 J=JBN,JTN
       IVGO=IV(I,J)+1
       JP1=J+1
       JM1=J-1
       GO TO (220,210,240,250,210,270,260,200,220,220,210) IVGO
210    V(I,J)=0.
       GO TO 200
220    DENOM=1.0+FAC(6)*SQRT(V(I,J)**2+UHAT(I,J)**2)/(D(I,J)+D(I,JM1))
       ELEV=FAC(4)*(H(I,J)-H(I,JM1))
       IF(MOD.EQ.0) THEN
          V(I,J)=(V(I,J)-FAC(5)*UHAT(I,J)-ELEV+DT*GV(I,J))/DENOM
       ELSE
          IF(IVGO-9) 221,222,223
221       UUHAT=(U(I,J)+U(IP1,J)+U(I,JM1)+U(IP1,JM1))/4.
          GO TO 224
222       UUHAT=(U(IP1,J)+U(IP1,JM1))/2.
          GO TO 224
223       UUHAT=(U(I,J)+U(I,JM1))/2.
224       V(I,J)=(V(I,J)-FAC(5)*UUHAT-ELEV+DT*GV(I,J))/DENOM
          UHAT(I,J)=UUHAT
       ENDIF
       GO TO 200
240    V(I,J)=-SQRT(G/D(I,J))*H(I,J)
       GO TO 200
250    V(I,J)=BB(I)
       GO TO 200
```

```
260   V(I,J)=BT(I)
      GO TO 200
270   V(I,J)=SQRT(G/D(I,JM1))*H(I,JM1)
200   CONTINUE
C
      IF (MOD.EQ.0) GO TO 12
11    CONTINUE
C
      IF(MOD.EQ.0) THEN
         DO 40 I=1,M
         IP1=I+1
         JBN=JB(I)
         JTN=JT(I)
         IF(JTN.EQ.0) GO TO 40
         DO 40 J=JBN,JTN
         IVGO=IV(I,J)+1
         GO TO (41,40,40,40,40,40,40,40,42,43,40) IVGO
41       JM1=J-1
         UHAT(I,J)=(U(I,J)+U(IP1,J)+U(I,JM1)+U(IP1,JM1))/4.
         GO TO 40
42       UHAT(I,J)=(U(IP1,J)+U(IP1,J-1))/2.
         GO TO 40
43       UHAT(I,J)=(U(I,J)+U(I,J-1))/2.
40       CONTINUE
      ELSE
         DO 50 I=1,M
         IM1=I-1
         JBN=JB(I)
         JTN=JT(I)
         IF(JTN.EQ.0) GO TO 50
         DO 50 J=JBN,JTN
         IUGO=IU(I,J)+1
         GO TO (51,50,50,50,50,50,50,50,52,53,50) IUGO
51       JP1=J+1
         VHAT(I,J)=(V(I,J)+V(I,JP1)+V(IM1,J)+V(IM1,JP1))/4.
         GO TO 50
52       JP1=J+1
         VHAT(I,J)=(V(I,JP1)+V(IM1,JP1))/2.
         GO TO 50
53       VHAT(I,J)=(V(I,J)+V(IM1,J))/2.
50       CONTINUE
      ENDIF
C
      RETURN
      END
```

Appendix 4

Program and Results for Sample Model

```
C   DRIVER FOR SAMPLE MODEL IN SECTION 10 OF REPORT 1.
C   USES QUADRATIC FRICTION STEPPING SUBROUTINE VQCA01
C   ALSO SUBROUTINES PRTB,PRINT LISTED BELOW.
C   USER MUST PROVIDE SUBROUTINE UFORCE (EXAMPLE
C   BELOW) TO SUPPLY FORCING AND BOUNDARY VALUES.
C
C  VALUES PRECEDED BY  * ARE SUPPLIED BY USER
C  VALUES PRECEDED BY ** ARE SUPPLIED BY USER VIA UFORCE
C
C  MODEL VARIABLES AND PARAMETERS
C
C **BB(I)       - BOTTOM BOUNDARY VALUE IN COLUMN I
C **BL(J)       - LEFT HAND BOUNDARY VALUE IN ROW J
C **BR(J)       - RIGHT HAND BOUNDARY VALUE IN ROW J
C **BT(I)       - TOP BOUNDARY VALUE IN COLUMN I
C  *D(I,J)      - WATER DEPTH AT H(I,J)                    (M)
C   DT          - TIME STEP                                (S)
C  *DX          - GRID INTERVAL IN X-DIRECTION             (M)
C  *DY          - GRID INTERVAL IN Y-DIRECTION             (M)
C  *F           - CORIOLIS COEFFICIENT                     (S**-1)
C  *G           - ACCELERATION DUE TO GRAVITY              (M/S**2)
C **GU(I,J)     - FORCING TERM AT U(I,J)                   (M/S**2)
C **GV(I,J)     - FORCING TERM AT V(I,J)                   (M/S**2)
C   H(I,J)      - SURFACE ELEVATION,ETA                    (M)
C   IE(I,J)     - INTERMEDIATE CODE AT H(I,J)
C   IU(I,J)     - INTERMEDIATE CODE AT U(I,J)
C   IV(I,J)     - INTERMEDIATE CODE AT V(I,J)
C   JB(I)       - LOWEST J VALUE PROCESSED IN COLUMN I
C   JT(I)       - HIGHEST J VALUE PROCESSED IN COLUMN I
C  *KE(I,J)     - PRIMARY CODE AT H(I,J)
C  *KU(I,J)     - PRIMARY CODE AT U(I,J)
C  *KV(I,J)     - PRIMARY CODE AT V(I,J)
C  *M           - NUMBER OF GRID MESHES IN X-DIRECTION
C   MP1         - M+1
C  *N           - NUMBER OF GRID MESHES IN Y DIRECTION
C   NP1         - N+1
C   NS          - NUMBER OF TIME STEP
C  *NSPC        - NUMBER OF STEPS PER CYCLE
C  *OMEGA       - TIDAL FREQUENCY                          (RAD/S)
C   T           - TIME                                     (S)
C   U(I,J)      - VELOCITY IN X-DIRECTION                  (M/S)
C   UHAT(I,J)   - SPACE-AVERAGED U-VELOCITY AT V(I,J)      (M/S)
C   V(I,J)      - VELOCITY IN Y-DIRECTION                  (M/S)
C   VHAT(I,J)   - SPACE-AVERAGED V-VELOCITY AT U(I,J)      (M/S)
C  *XK          - QUADRATIC FRICTION COEFFICIENT,K
C
C
C  CONTROL PARAMETERS FOR RUN
```

```
C
C   *ICODES     - .EQ.0 READ PRIMARY CODES FROM UNIT NO. KEUVD
C                 .NE.0 READ INTERMEDIATE CODES FROM UNIT NO. IEUVD
C   *IRSTRT     - .EQ.0 ZERO INITIAL CONDITIONS
C                 .EQ.1 READ RESTART DATA FROM UNIT NO. INPT
C                 .EQ.2 READ RESTART DATA FROM UNIT NO. INPT
C                       AND RESET T=0 , NS=0
C   *NCT        - NO. OF STEPS IN THIS RUN
C   *NLPST      - NO. OF STEP IN THIS RUN AT WHICH OUTPUT BEGINS
C   *NPT        - STORE RESTART DATA ON UNIT NO. IOUT
C                 EVERY NPT STEPS FOR NPT.GE.NLPST
C   *NREC       - RECORD NO. OF RESTART DATA ON UNIT NO. INPT
C
C   I/O UNIT NUMBERS
C
C   *IEUVD      - UNIT NO. FOR FILE WITH IE,IU,IV,JB,JT,D
C   *KEUVD      - UNIT NO. FOR FILE WITH KE,KU,KV,D
C   *INPT       - UNIT NO. FOR INPUT OF RESTART DATA
C   *IOUT       - UNIT NO. FOR OUTPUT OF RESTART DATA
C   *ILINE      - UNIT NO. FOR LINEPRINTER OR EQUIVALENT
C   *IREAD      - UNIT NO. FOR CARDREADER OR EQUIVALENT
C
      DIMENSION  H(7,5),     U(8,5),    V(7,6)
      DIMENSION KE(7,5),    KU(8,5),   KV(7,6)
      DIMENSION IE(7,5),    IU(8,5),   IV(7,6)
      DIMENSION  D(7,5), UHAT(8,5),VHAT(7,6)
      DIMENSION             GU(8,5),   GV(7,6)
      DIMENSION JB(8),JT(8),BB(7),BT(7),BR(5),BL(5)
C
      IREAD=5
      READ(IREAD,121)INPT
      READ(IREAD,121)IOUT
      READ(IREAD,121)ILINE
      READ(IREAD,121)IEUVD
      READ(IREAD,121)KEUVD
      READ(IREAD,121)ICODES
      READ(IREAD,121)IRSTRT
      READ(IREAD,121)NCT
      READ(IREAD,121)NPT
      READ(IREAD,121)NLPST
      READ(IREAD,121)NREC
      READ(IREAD,121)NSPC
      READ(IREAD,121)M
      READ(IREAD,121)N
      READ(IREAD,122)DX
      READ(IREAD,122)DY
      READ(IREAD,122)F
      READ(IREAD,122)G
      READ(IREAD,122)OMEGA
      READ(IREAD,122)XK
  121 FORMAT(I5)
  122 FORMAT(F15.8)
      MP1=M+1
      NP1=N+1
      PI=3.14159265
```

```
      DT=2.*PI/(OMEGA*NSPC)
C
      WRITE(ILINE,100) M,N,DT,DX,DY,G,F,XK,OMEGA,NCT
100   FORMAT('1NO. OF MESHES IN X-DIRECTION =',I5/1X,
     *       'NO. OF MESHES IN Y-DIRECTION =',I5/1X,
     *       'TIME STEP = ',F14.6/,1X,
     *       'GRID INTERVAL IN X-DIR (M) = ',F9.2/,1X,
     *       'GRID INTERVAL IN Y-DIR (M) = ',F9.2/,1X,
     *       'ACCELERATION DUE TO GRAVITY = ',F9.5/,1X,
     *       'CORIOLIS COEFFICIENT = ',E12.4/,1X,
     *       'QUADRATIC FRICTION COEFFICIENT = ',E12.4/,1X,
     *       ' TIDAL FREQUENCY = ',E15.8/,1X,
     *       'NUMBER OF TIME STEPS IN THIS RUN = ',I8/)
C
      IF(ICODES.EQ.0) THEN
C
C   READ GRID CODES FROM UNIT #KEUVD
C
          READ(KEUVD,111) ((KE(I,J),I=1,M),J=N,1,-1)
          READ(KEUVD,110) ((KU(I,J),I=1,MP1),J=N,1,-1)
          READ(KEUVD,111) ((KV(I,J),I=1,M),J=NP1,1,-1)
          READ(KEUVD,112) ((D(I,J),I=1,M),J=N,1,-1)
          CALL CLOSE(KEUVD,1)
110       FORMAT(8I2)
111       FORMAT(7I2)
112       FORMAT(7F6.0)
          CALL PRT(D,ILINE,KE,KU,KV,M,MP1,N,NP1)
C
C   CALL SUBROUTINE TO CONVERT PRIMARY CODES TO INTERMEDIATE CODES
C
          CALL BICAD1(IE,IU,IV,JB,JT,KE,KU,KV,M,MP1,N,NP1)
C
C
C   WRITE INTERMEDIATE CONDITIONS ON UNIT # IEUVD
C
          WRITE (IEUVD)IU,IV,IE,JB,JT,D
          CALL CLOSE (IEUVD,1)
          WRITE(ILINE,130)
          CALL PRTB(D,ILINE,IE,IU,IV,JB,JT,M,MP1,N,NP1)
          WRITE(ILINE,129)
C
      ELSE
C
C   READ INTERMEDIATE DATA ON UNIT # IEUVD
C
          READ(IEUVD) IU,IV,IE,JB,JT,D
          CALL CLOSE (IEUVD,1)
          WRITE(ILINE,131)IEUVD
          CALL PRTB(D,ILINE,IE,IU,IV,JB,JT,M,MP1,N,NP1)
      ENDIF
      IF(IRSTRT.EQ.0) THEN
C
C  ZERO INITIAL CONDITIONS
C
          DO 1 I=1,M
```

```
          DO 1 J=1,N
          H(I,J)=0.
          U(I,J)=0.
          V(I,J)=0.
          UHAT(I,J)=0.
          VHAT(I,J)=0.
1         CONTINUE
          DO 2 I=1,M
          V(I,NP1)=0.
          VHAT(I,NP1)=0.
2         CONTINUE
          DO 3 J=1,N
          U(MP1,J)=0.
          UHAT(MP1,J)=0.
3         CONTINUE
          NSO=0
          WRITE(ILINE,142)
      ELSE
C
C   NON-ZERO INITIAL CONDITIONS AVAILABLE
C
C   READ INITIAL CONDITIONS ON UNIT NO. INPT
C
          DO 10 J=1,NREC
          READ(INPT) H,U,V,NSO,T
10        CONTINUE
          CALL CLOSE (INPT,1)
C   FOLLOWING DUMMY STEP WITH DT=0 INITIALIZES UHAT,VHAT
          CALL UFORCE(BB,BL,BR,BT,GU,GV,M,MP1,N,NP1,T)
          CALL VOCAD1(BB,BL,BR,BT,0,0.,DX,DY,F,G,GU,GV,H,IE,IU,IV,
     1    JB,JT,M,MP1,N,NP1,NSO,T,U,UHAT,V,VHAT,XK)
          IF(IRSTRT.EQ.2) THEN
             NSO=0
             T=0.
          ENDIF
          WRITE(ILINE,141) NSO,T
          CALL PRINT(H,ILINE,IE,IU,IV,M,MP1,N,NP1,U,V)
      ENDIF
C
129   FORMAT('1')
130   FORMAT(//'1INTERMEDIATE CONDITIONS AFTER SUBROUTINE BICAD1'//)
131   FORMAT(//'1INTERMEDIATE CONDITIONS READ FROM UNIT #',I6//)
C
      NS=NSO+1
C
C   MAIN COMPUTING LOOP
C
20    T=NS*DT
C
      CALL UFORCE(BB,BL,BR,BT,GU,GV,M,MP1,N,NP1,T)
      CALL VOCAD1(BB,BL,BR,BT,0,DT,DX,DY,F,G,GU,GV,H,IE,IU,IV,
     1JB,JT,M,MP1,N,NP1,NS,T,U,UHAT,V,VHAT,XK)
C
C   PRINTOUT AND SAVING OF VALUES ON UNIT # IOUT
C
```

```
      NNS=NS-NSO
      IF(NNS.GE.NLPST) THEN
         NNSS=NNS-NLPST
         MOD=NNSS-(NNSS/NPT)*NPT
         IF(MOD.EQ.0) THEN
            WRITE(ILINE,140)NNS, NS,T
            WRITE(IOUT) H,U,V,NS,T
            CALL PRINT(H,ILINE,IE,IU,IV,M,MP1,N,NP1,U,V)
         ENDIF
      ENDIF
C
140   FORMAT(//' STEP NO.(THIS RUN)=',I8/' CUMULATIVE STEP NO. =',I8/
     1' TIME SINCE T=0 IN SECONDS IS',E20.6)
141   FORMAT(//' STEP NO.(RESTART)=',I8/' TIME SINCE T=0 IN SECONDS IS'
     *E20.6)
142   FORMAT(//' ZERO INITIAL CONDITIONS')
C
      NS=NS+1
      IF(NS-NSO.LE.NCT) GO TO 20
      END



      SUBROUTINE PRTB(D,ILINE,KE,KU,KV,JB,JT,M,MP1,N,NP1)
C
      DIMENSION KU(MP1,N),KV(M,NP1),KE(M,N)
      DIMENSION JB(MP1),JT(MP1),D(M,N)
C
      WRITE(ILINE,1)
      WRITE(ILINE,2) (JB(I),I=1,MP1)
      WRITE(ILINE,3)
      WRITE(ILINE,2) (JT(I),I=1,MP1)
1     FORMAT(/' LOWER BOUNDS FOR EACH COLUMN  - JB'/)
3     FORMAT(//' UPPER BOUNDS FOR EACH COLUMN  - JT'/)
2     FORMAT(20I4//)
      ENTRY PRT(D,ILINE,KE,KU,KV,M,MP1,N,NP1)
      WRITE(ILINE,10)
      WRITE(ILINE,11)(J,(D(I,J),I=1,M),J=N,1,-1)
      WRITE(ILINE,16)
      WRITE(ILINE,15)(J,(KE(I,J),I=1,M),J=N,1,-1)
      WRITE(ILINE,12)
      WRITE(ILINE,13)(J,(KU(I,J),I=1,MP1),J=N,1,-1)
      WRITE(ILINE,14)
      WRITE(ILINE,15)(J,(KV(I,J),I=1,M),J=NP1,1,-1)
C
10    FORMAT(//15X,' DEPTHS'/'  J')
12    FORMAT(//15X,' U VELOCITY CODE'/'  J')
14    FORMAT(//15X,' V VELOCITY CODE'/'  J')
16    FORMAT(//15X,' ELEVATION CODE'/'  J')
11    FORMAT(1X,I2,10X,7F5.0)
13    FORMAT(1X,I2,10X,8I4)
15    FORMAT(1X,I2,10X,7I4)
      RETURN
      END
```

```
      SUBROUTINE PRINT(H,ILINE,IE,IU,IV,M,MP1,N,NP1,U,V)
C
      DIMENSION H(M,N),U(MP1,N),V(M,NP1),IE(M,N),IU(MP1,N),IV(M,NP1)
      CHARACTER LINE*12(10)
C
      WRITE(ILINE,1)
1     FORMAT(/' SURFACE ELEVATION')
      DO 10 J=N,1,-1
      DO 11 I=1,M
      IF(IE(I,J).EQ.7) THEN
         LINE(I)='            '
      ELSE
         ENCODE(12,100,LINE(I)) H(I,J)
      ENDIF
11    CONTINUE
      WRITE(ILINE,110) (LINE(K),K=1,M)
10    CONTINUE
C
100   FORMAT(F12.7)
110   FORMAT(10A12)
111   FORMAT(10A12)
C
      WRITE(ILINE,2)
2     FORMAT(/' U VELOCITY COMPONENT')
      DO 20 J=N,1,-1
      DO 21 I=1,MP1
      IF(IU(I,J).EQ.7) THEN
         LINE(I)='            '
      ELSE
         ENCODE(12,100,LINE(I)) U(I,J)
      ENDIF
21    CONTINUE
      WRITE(ILINE,111) (LINE(K),K=1,MP1)
20    CONTINUE
C
      WRITE(ILINE,3)
3     FORMAT(/' V VELOCITY COMPONENT')
      DO 30 J=NP1,1,-1
      DO 31 I=1,MP1
      IF(IV(I,J).EQ.7) THEN
         LINE(I)='            '
      ELSE
         ENCODE(12,100,LINE(I)) V(I,J)
      ENDIF
31    CONTINUE
      WRITE(ILINE,110) (LINE(K),K=1,M)
30    CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE UFORCE(BB,BL,BR,BT,GU,GV,M,MP1,N,NP1,T)
C
C   COMPUTES FORCING TERMS AND BOUNDARY VALUES FOR SAMPLE
C   MODEL IN REPORT I.
C
      DIMENSION GU(MP1,N),GV(M,NP1),BB(M),BT(M),BL(N),BR(N)
      OM=1.4052E-4
      DX=6000.
      DY=5000.
      C1=-3.76E-4
      C2=2.17E-4
      C3=4.78E-7
      C4=-2.76E-7
      C3DX=C3*DX
      C4DY=C4*DY
      OMT=OM*T
      DO 100 I=1,MP1
      DO 100 J=1,N
      ARG=OMT+C3DX*(I-1)+C4DY*(J-0.5)
      IF(ARG.LE.0.)GU(I,J)=0.
      IF(ARG.GT.0.)GU(I,J)=C1*SIN(ARG)
  100 CONTINUE
      DO 101 I=1,M
      DO 101 J=1,NP1
      ARG=OMT+C3DX*(I-0.5)+C4DY*(J-1)
      IF(ARG.LE.0.)GV(I,J)=0.
      IF(ARG.GT.0.)GV(I,J)=C2*SIN(ARG)
  101 CONTINUE
      SOMT=SIN(OMT)
      BL(1)=0.21*SOMT
      BL(2)=0.22*SOMT
      BL(3)=0.23*SOMT
      ARG=OMT-0.135
      IF(ARG.LE.0.)BR(2)=0.
      IF(ARG.GT.0.)BR(2)=0.88*SIN(ARG)
      ARG=OMT-0.138
      IF(ARG.LE.0.)BR(3)=0.
      IF(ARG.GT.0.)BR(3)=0.89*SIN(ARG)
      ARG=OMT-0.140
      IF(ARG.LE.0.)BR(4)=0.
      IF(ARG.GT.0.)BR(4)=0.90*SIN(ARG)
      ARG=OMT-0.077
      IF(ARG.LE.0.)BB(4)=0.
      IF(ARG.GT.0.)BB(4)=0.12*SIN(ARG)
      ARG=OMT-0.096
      IF(ARG.LE.0.)BB(5)=0.
      IF(ARG.GT.0.)BB(5)=0.13*SIN(ARG)
      RETURN
      END
```

Output from Sample Model

```
NO. OF MESHES IN X-DIRECTION =     7
NO. OF MESHES IN Y-DIRECTION =     5
TIME STEP =        46.001867
GRID INTERVAL IN X-DIR (M) =    6000.00
GRID INTERVAL IN Y-DIR (M) =    5000.00
ACCELERATION DUE TO GRAVITY =    9.81000
CORIOLIS COEFFICIENT =     .1200-003
QUADRATIC FRICTION COEFFICIENT =     .2500-002
NUMBER OF TIME STEPS IN THIS RUN =        23
```

DEPTHS

J							
5	0.	0.	0.	167.	165.	0.	0.
4	0.	0.	111.	155.	164.	156.	153.
3	172.	174.	161.	145.	156.	145.	148.
2	151.	162.	150.	123.	125.	132.	130.
1	120.	91.	83.	90.	95.	0.	0.

ELEVATION CODE

J							
5	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1
3	0	0	0	0	0	0	1
2	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0

U VELOCITY CODE

J								
5	0	0	0	1	0	1	0	0
4	0	0	1	0	0	0	0	0
3	3	0	0	0	0	0	0	0
2	3	0	0	0	0	0	0	0
1	3	0	0	0	0	1	0	0

V VELOCITY CODE

J							
6	0	0	0	4	4	0	0
5	0	0	1	0	0	1	1
4	1	1	0	0	0	0	0
3	0	0	0	0	0	0	0
2	0	2	0	0	0	1	1
1	1	1	1	3	3	0	0

INTERMEDIATE CONDITIONS AFTER SUBROUTINE BICA01

LOWER BOUNDS FOR EACH COLUMN - JB

1	1	1	1	1	1	2	0

UPPER BOUNDS FOR EACH COLUMN - JT

3	3	4	5	5	5	4	0

ELEVATION CODE

J							
5	7	7	7	0	0	7	7
4	7	7	0	0	0	0	2
3	0	0	0	0	0	0	2
2	0	0	0	0	0	0	2
1	0	0	0	0	0	7	7

U VELOCITY CODE

J								
5	7	7	7	1	0	4	7	7
4	7	7	1	0	0	0	0	7
3	3	0	0	0	0	0	0	7
2	3	0	0	0	0	0	0	7
1	3	0	0	0	0	4	7	7

V VELOCITY CODE

J							
6	7	7	7	5	5	7	7
5	7	7	4	0	0	4	4
4	4	4	0	0	0	0	9
3	0	0	0	0	0	0	9
2	0	10	0	0	0	1	1
1	1	1	1	3	3	7	7

Output from Sample Model

```
STEP NO.(THIS RUN)=       22
CUMULATIVE STEP NO. =       22
TIME SINCE T=0 IN SECONDS IS          .101204+004

SURFACE ELEVATION
                                      .0627580    .0328530
                          .1084765    .0634346    .0326364    .0189228    .0019908
   .2050225    .1520726   .1011221    .0568959    .0235798    .0074186    .0037487
   .1907428    .1372952   .0897075    .0458161    .0092818   -.0049634    .0063465
   .1780325    .1287519   .0796102    .0365285   -.0077946

U VELOCITY COMPONENT
                                      .0000000   -.0079399    .0000000
                          .0000000    .0011358   -.0115892   -.0272206   -.0215388
   .0325986    .0214209   .0134374    .0010636   -.0122066   -.0265824   -.0319199
   .0311813    .0203030   .0122392    .0007600   -.0117187   -.0312812   -.0440566
   .0297640    .0232174   .0096040   -.0002465   -.0045940    .0000000

V VELOCITY COMPONENT
                                      .0152106    .0080106
                          .0000000    .0131870    .0177976    .0000000    .0000000
   .0000000    .0000000   .0071921    .0076848    .0086621    .0080533    .0208639
  -.0027758   -.0006540   .0013311    .0020851    .0017138    .0074246    .0217405
  -.0033964    .0000000  -.0021981   -.0005406   -.0089424    .0000000    .0000000
   .0000000    .0000000   .0000000    .0078199    .0060054
```

STEP NO.(THIS RUN)= 23
CUMULATIVE STEP NO. = 23
TIME SINCE T=0 IN SECONDS IS .105804+004

SURFACE ELEVATION

			.0690262	.0375230		
		.1163175	.0698371	.0376712	.0221956	.0078085
.2154740	.1622176	.1100363	.0641935	.0296320	.0109083	.0095016
.2023710	.1480286	.0992903	.0546522	.0176938	-.0009933	.0120347
.1908708	.1411255	.0885360	.0466278	.0032464		

U VELOCITY COMPONENT

			.0000000	-.0081477	.0000000	
		.0000000	.0020437	-.0117802	-.0287331	-.0231901
.0340697	.0228693	.0147716	.0018865	-.0122784	-.0278861	-.0345303
.0325884	.0218047	.0132778	.0014444	-.0116695	-.0326463	-.0478151
.0311071	.0243542	.0109079	.0002151	-.0040716	.0000000	

V VELOCITY COMPONENT

			.0167298	.0091493		
		.0000000	.0148080	.0194294	.0000000	.0000000
.0000000	.0000000	.0081079	.0087401	.0096129	.0087856	.0227971
-.0026464	-.0005411	.0018401	.0028029	.0023316	.0081544	.0238309
-.0031067	.0000000	-.0016686	.0003187	-.0085871	.0000000	.0000000
.0000000	.0000000	.0000000	.0085938	.0068447		

Amplitude and Phase of Constituent with Frequency

$\omega = 1.4052 \cdot 10^{-4}\ s^{-1}$ during 11th Cycle of Sample Model

SURFACE ELEVATION - AMPLITUDE (M)

			1.7350	1.5325		
		1.8498	1.6320	1.4203	1.1827	.9000
2.2125	1.9810	1.7538	1.5353	1.3196	1.1014	.8900
2.1164	1.8856	1.6580	1.4368	1.2223	1.0217	.8800
2.0197	1.7898	1.5591	1.3342	1.1146		

SURFACE ELEVATION - PHASE (RAD.)

			-.0534	-.0618		
		-.0209	-.0399	-.0525	-.0661	-.1400
.0092	-.0001	-.0146	-.0314	-.0481	-.0745	-.1380
.0098	-.0000	-.0117	-.0254	-.0409	-.0665	-.1350
.0112	.0032	-.0099	-.0206	-.0322		

U VELOCITY - AMPLITUDE (M/S)

			.0000	.0289	.0000	
		.0000	.2746	.0948	.3158	.8959
.2300	.2396	.2774	.2122	.0661	.1154	.4275
.2200	.2350	.2058	.1543	.0785	.2540	.9498
.2100	.1897	.2307	.1094	.0554	.0000	

U VELOCITY - PHASE (RAD.)

			.0000	-1.8153	.0000	
		.0000	-.2158	-1.1724	-1.9159	-1.3335
.0000	-.0720	-.1378	-.1925	-.5822	-2.7679	-2.9187
.0000	-.0674	-.0728	-.0424	.5198	1.5557	1.6063
.0000	.0053	-.0699	-.1001	.0765	.0000	

V VELOCITY - AMPLITUDE (M)

			.4205	.3737		
		.0000	.4318	.3804	.0000	.0000
.0000	.0000	.2221	.3041	.3162	.5500	1.5566
.0108	.0247	.1182	.1962	.2459	.5578	1.5948
.0339	.0000	.0769	.1373	.1510	.0000	.0000
.0000	.0000	.0000	.1200	.1300		

V VELOCITY - PHASE (RAD.)

			-.0534	-.0618		
		.0000	-.0923	.0183	.0000	.0000
.0000	.0000	-.1736	-.2291	-.5365	-1.0468	-1.0974
-.5647	-.3545	-.2107	-.3358	-.8374	-1.5184	-1.5664
-.3342	.0000	-.1655	-.1984	-.0989	.0000	.0000
.0000	.0000	.0000	-.0770	-.0960		