AUTOMATED DIGITAL SIMULATION OF TIDES & LONG WAVES

R.F. Henry*

Abstract

A system for indirect programming of shallow-water models is discussed, particular attention being given to facilitating successful operation by novice users. Pre-developed programs are used to check numerical coding of model layout, to execute the finite-difference computations required at each time step and to analyse the computed surface elevations and velocities. Many error-prone steps are thus eliminated and model development is speeded for modest increase in computation costs.

1. Introduction

This paper describes a method which simplifies development of numerical models of long-wave phenomena such as tides, tsunamis and storm surges. The irregular geometry of most coastal seas complicates the programming of numerical models severely, even for the simplest finite-difference representations of the governing shallow-water equations. The principal aim of the system of programs described here is to automate the detailed programming steps necessary to fit a finite-difference scheme to a particular coast. The availability of such programs permits even an inexperienced programmer to develop a working model quickly and, at the subsequent calibration and verification stages, allows experimental changes, such as relocation of boundaries, to be made rapidly without extensive debugging checks. The modeller is thus freed from programming difficulties and delays and is able to concentrate on scientific aspects of the work.

The basic concept used is that, after a suitable grid has been laid out to cover the coast in question, instead of programming the specific computational steps necessary for each variable on the grid, a numerical code is assigned to each variable to indicate the nature of that point and the type of calculation to be carried out there at each time step. In this way the task of programming the finite-difference calculations is split into two distinct stages,

- a) numerical coding of coastline layout
- b) development of a general time-stepping subroutine, which, given the corresponding code, performs the computation required at each grid-point under the particular finite-difference scheme adopted.

The advantage of this approach is that step b) has to be tackled

* Institute of Ocean Sciences, Sidney, B.C., Canada.

only once for a particular finite-difference scheme, and when the time-stepping subroutine has been fully checked out, a major source of programming errors is eliminated from all models employing it subsequently. Provided that the same governing equations hold, only stage a) has to be carried out when a new model is developed and as shown later, a graphical checking procedure can be used to ensure that no errors are made at that stage.

Leendertse (1967) used numerical coding to specify location of model boundaries but Heaps (1969) seems to have been first to perform all computations on the grid indirectly via numerical codes. Abbott, Damsgaard and Rodenhuis (1973) announced an extensive proprietary software package for indirect programming of shallow-water models, but limited information about this system and its later developments has appeared in the open literature. The work described in the present paper essentially extends Heaps' method, increasing the practicability by using a simpler grid, by breaking the coding process down into stages of which only the simplest has to be carried out by the user and by providing graphical means for checking the coding. The result is a set of programmers but which are straightforward enough to be altered fairly readily by more experienced users to suit special requirements.

Unfortunately, the combination of grid and difference-scheme used by Heaps proves rather cumbersome for indirect programming. Because both components are defined at all velocity-points, 19 distinct types of velocity point are possible, even though only two types of boundary are permitted. This complexity makes it difficult to code the model geometry correctly. The Richardson grid (Figure 1) used here is better suited to indirect programming, as far fewer distinct configurations can occur at boundaries. This has facilitated development of a plotting program to permit visual checking of coded model layout. A plot of the model boundaries is produced from the allotted numerical codes. Comparison with the original grid diagram then quickly reveals any errors in coding. If the model boundaries are altered during development, the revised layout can be checked similarly. This graphical feedback makes indirect programming considerably more practicable.

To summarize the procedure, the modeller is required to choose a Cartesian grid of suitable orientation, size and spacing for the water body in question and to provide a corresponding array of mean water depths. The position and nature of the various boundaries are then coded numerically as described later and the coding is checked visually using a pre-developed graphical program. The numerical model is then programmed, all the necessary finite-difference calculations being handled by a pre-developed time-stepping subroutine. The modeller must supply a subroutine which provides values of any time-dependent forces or boundary conditions influencing the water body.

A model programmed using this technique takes roughly 20% more computing time than an equivalent model in which the finite-difference equations are programmed specifically for the particular grid in



Fig. 1. The Richardson Grid

question, but the saving in initial programming effort and subsequent debugging time is substantial even for an inexperienced programmer. Memory requirements are increased by the need to store the code arrays required. Full details of the various programs which have been developed to implement the graphical and finite-difference operations discussed above are given in Henry (1982).

2. Governing Equations

The methods described in this paper apply to problems governed by the partly linearized shallow water equations

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

$$u_t = -g_{n_X} + f_V - F^{(u)} + G^{(u)}$$
 (2)

$$v_t = -g_{n_v} - f_u - F^{(v)} + G^{(v)}$$
 (3)

where

η(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 bottom friction terms. One stepping subroutine makes provision for friction linearly proportional to velocity, i.e. F(u) = ru, F(v) = rv, while another uses the more commonly assumed quadratic friction forms:

$$F(u) = \frac{ku(u^2 + v^2)^{1/2}}{d} ; F(v) = \frac{kv(u^2 + v^2)^{1/2}}{d}$$
(4)

The respective friction coefficients r and k are currently taken to be uniform throughout the model, but later versions of the stepping subroutines will permit the friction coefficient to vary with position. Also it is intended to supply a version using (d^4n) in place of d in the denominators of (4), in order to simulate fluid behavior in very shallow water more accurately. By design, these stepping subroutines are readily interchangeable, so that variations on the underlying finite-difference scheme can be tried in the same model with negligible reprogramming. It would be possible to write a master stepping subroutine containing all options likely to be required, but this would be less economical in computing time. The terms G(u) and G(v) in equations (1) - (3) represent relevant forcing effects due to external influences. In general these will be both time- and space-dependent. Appropriate values for these must be specified for the particular problem considered through a user-provided subroutine.

3. The Richardson-Sielecki Finite-Difference Scheme

.

The simple Richardson grid shown in Figure 1 was chosen as the basis for the finite-difference scheme because it minimizes storage requirements and permits particularly simple simulation of coastlines. At interior points of the grid, equations (1) to (3) are represented by Sielecki's (1968) scheme.

$$\frac{n_{ij} - n_{ij}}{\Delta t} = -\frac{(d_{ij} + d_{i+1,j})u_{i+1,j} - (d_{i-1,j} + d_{ij})u_{ij}}{2.\Delta x} - \frac{(d_{ij} + d_{i,j+1})v_{i,j+1} - (d_{i,j-1} + d_{ij})v_{ij}}{2.\Delta y}$$
(5)

$$\frac{u_{ij} - u_{ij}}{\Delta t} = -g \frac{u_{ij} - u_{i-1,j}}{\Delta x} + f\tilde{v}_{ij} - F(u) + G(u) + G(u)$$
(6)

$$\frac{v_{ij} - v_{ij}}{\Delta t} = -g - \frac{n_{ij} - n_{i,j-1}}{\Delta y} - f\tilde{u}_{ij}' - F(v) + G(v) + G(v)$$
(7)

where

.

 Δt = time step $\Delta x, \Delta y$ = grid interval sizes in in x,y directions respectively d_{ii} = mean water depth at elevation point n_{ii}

$$\tilde{u}_{ij} = \frac{1}{4} \left[u_{i,j-1} + u_{1+1,j-1} + u_{ij} + u_{i+1,j} \right]$$
$$\tilde{v}_{ij} = \frac{1}{4} \left[v_{i-1,j} + v_{ij} + v_{i-1,j+1} + v_{i,j+1} \right]$$

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 (6) and new (primed) values of u in the corresponding term in (7) 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 n_{ij} are updated, then all of the u_{ij} , and finally all the v_{ij} . The same conclusion applies if

evaluated in the order n_1 , 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 provided evaluate the variables in the order n', u', v', on odd-numbered steps and n', v', u', on even-numbered steps.

Sielecki showed that the condition for stability of the above scheme, in the absence of boundaries, is

$$\Delta t \leq \frac{\Delta x \cdot \Delta y}{\left[gd_{max}(\Delta x^2 + \Delta y^2)\right]^{1/2}}$$
(8)

4. Numerical Coding of Model Geometry

The choice of grid for a model is necessarily a compromise among many requirements. The location and orientation of the grid is influenced mainly by the accuracy with which the coastlines can be approximated by line segments of the grid. This fit can usually be improved by increasing the grid resolution, which implies using more variable points and taking shorter time steps to maintain numerical stability, thus raising computing costs. Once the grid has been chosen, as objectively as possible, all variable points on the grid are allotted appropriate (primary) integer codes. An elevation point nij is given the code number $(\text{KE})_{ij} = 1$ if the point lies on a sea-boundary along which surface elevation point has the code $(\text{KE})_{ij} = 0$.

Points on the grid where velocity components $u_{i\,j}$ or $v_{i\,j}$ are defined are each allotted a corresponding subscripted code (KU)_{i\,j} or (KV)_{i\,j} which can have the following values:

Primary codes KU,KV at Velocity Points Location of Velocity Point

1	on line segment representing land boundary
2	on line segment representing spit or causeway with water on both sides
3	at sea boundary where velocity is to be
4	on radiating sea boundary (discussed below)
0	elsewhere

The allocation of these codes will now be illustrated with the aid of the very simple example in Figure 2(a). Here, the velocities u_1 , u_{12} , u_{13} on the sea-boundary at the left side and v_{41} , v_{51} at the bottom edge of the grid are specified functions of time derived from current meter observations. Elevations n_{72} , n_{73} , n_{74} at the right-hand sea boundary are also known functions of time, obtained from water level gauges.

The upper boundary on which points $v_{46}^{}, v_{56}^{}$ are located is a



Fig. 2. Sample Water Body and Primary Codes

radiating sea boundary. 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 2). 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

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:

or

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

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

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

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

or

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

For example, in the model shown in Figure 2, at each time step, the new velocity values v'_{46} , v'_{56} , on the radiating boundary are found from the newly-updated elevations n'_{45} , n'_{55} by putting

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

This simple but effective radiation condition was introduced by Heaps (1974). 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°. Tests show that use of the radiation condition can reduce the permissible time step by as much as 50%. Equation (8) should therefore be amended to:

$$\Delta t \leq \frac{\Delta x \cdot \Delta y}{2[gd(\Delta x^2 + \Delta y^2)]^{1/2}}$$



Fig. 3. Preparatory Steps

to ensure stability in models using the above type of radiating boundary.

There is to be no flow across any 'land' boundary in the model and consequently, all the corresponding velocity components, e.g. u_{34} , u_{61} , v_{11} , v_{24} , are to be set at zero. Also, v_{22} is zero, as there can be no transverse flow at the line segment representing the small island where v_{22} is situated.

The arrays of numerical codes which describe the geometry of this model, according to the list given above, are shown in Figures 2(b),(c) and (d). This coding stage of the procedure is represented by box A in Figure 3, which shows the sequence of preparatory steps required for a model. Currently, a file containing the numerical code arrays has to be prepared by the modeller, but work is in progress using interactive graphics (E in Figure 3) to replace manual entry. The coastlines and superposed grid are displayed and the grid segments which are to represent boundaries are then selected and designated using a cursor or light pen. The nature of each boundary is entered at the same time through the keyboard. However, manual entry of the primary codes, which will remain necessary in the absence of suitable graphics hardware, is very straightforward, due to the limited number of codes involved and the provision of a program for graphical checking. The latter requires only an off-line graphics facility.

5. Graphical Check of Primary Codes

Correct description of the location and types of the model boundaries is of such obvious importance that a program has been developed for plotting the model layout defined by the allocated primary codes. The plotting scale used can be set equal to that of the chart on which the model grid was set out, so that by overlaying the plot on the chart the correctness of the numerical coding can be checked very readily. It is also worthwhile repeating this graphical check (C in Figure 3) whenever changes are made to the primary codes, for whatever reason.

Figure 4 shows the plot thus produced from the primary codes given in Figures 2 (b), (c) and (d). The boundaries are distinguished by use of the following line codes:

 closed boundaries
 spit or causeway
 sea boundary with specified variable
 radiating sea boundary

The plotting program uses standard CALCOMP plotting subroutines.

6. Conversion from Primary to Intermediate Codes

Although the primary codes describe fully the location and type of every boundary, they do not contain certain information about relative locations required by the time-stepping finite-difference



Fig. 4. Layout of Sample Model as Plotted from Primary Codes

subroutines. For this reason, another program is used (B in Figure 3) to convert the primary codes into a more numerous set of 'intermediate' codes. The latter distinguish, for example, between inactive variable points on land (or outside the model domain) and active points in the model interior. The primary to intermediate conversion program also works out the bounds of the area of grid containing active variable points, thus reducing the amount of computation required in the time-stepping subroutines at each time-step. Users generally need not concern themselves with details of the intermediate codes and scanning bounds.

Boxes G and H and Fig. 3 refer to programs being developed to permit substitution of a stepping subroutine based on the full non-linear shallow-water equations in place of the partly linearized versions presently available. An even fuller set of codes, here termed 'secondary', are required in that case to cope with the variety of special cases which occur in the vicinity of boundaries.

7. Programming a Model

After the above preparatory stages of coding, checking and code conversion have been carried out, one proceeds very much as in programming a model by the usual direct method, except that the intricate programming of the finite-difference calculations is replaced by a simple call to the appropriate time-stepping subroutine. The other major difference is that time-dependent forces and boundary values are supplied at each time-step by calling a subroutine which loads specific storage locations with this data, rather then by summoning values directly where required in the finite-difference coding.

It is useful to visualize the linear arrays used to store boundary values as lying parallel to the boundaries on which the values are used, as shown in Figure 5, which refers to the simple example of Figure 2. By having each array the same size as its respective side of the model grid, the same numbering can be used for both. Thus in the example, the values to be supplied at each step for u_{1j} are stored in (BL)_j, j=1,2,3; those for v_{11} , in (BB)_i, i=4,5 and those for n_{7j} in (BR)_j, j=2,3,4. The storage arrays, which are a fixed feature of the time-stepping subroutines, can be used for holding elevation or velocity values as required for the model in question. Cases where this storage arrangement could lead to values for two different boundary points being allotted to the same storage location hardly ever seem to occur in practice and have not been catered for. Values supplied by the user at each time-step for the forcing terms $G^{(u)}$ and $G^{(v)}$ (equations 2 and 3) at every relevant variable point are stored in two-dimensional arrays where they are subsequently accessed by the stepping subroutine.

Some of the storage locations just discussed go unused in most models - for example, none of the boundary storage array BT pertaining to the upper boundary in Figure 5 is used - but this is a reasonable price to pay for other conveniences of the method.



Fig. 5. Storage of Boundary Input Values for Sample Model



Fig. 6. Typical Model Flowchart

Figure 6 shows a simplified flowchart suitable for many models. The conversion from primary to intermediate code is shown included in the model program, since in many cases, modellers make numerous boundary changes during model development. Some further details are given in Henry (1982) and the results of an actual run with the model shown in Figure 2 are listed there for check-out purposes.

8. Further Developments

Besides extensions already mentioned, a number of auxiliary programs are being developed, mainly for performing standard analyses of results from production runs with the completed model. For tidal studies there is a program which carries out harmonic analysis of computed currents and surface elevations, while for storm surge models another program is being developed to log the maximum height reached at each elevation point during any specified period. Since the model shape is already expressed in the primary codes, the latter are used to confine the analysis computations to active variable points, to organize the layout of printed results of the analyses, and also to produce plots of the boundaries in subsequent graphical programs, such as one which can contour any designated scalar field computed during the analysis.

9. Conclusions

The method of programming shallow water models discussed in this paper is designed to answer a need for rapid, error-free application of well-known finite-difference methods to different coastal areas. The basic concept involved, indirect programming or selection of algorithms via numerical codes, has attracted interest for a number of years but has not come into widespread use. This paper proposes some improvements which should help to make indirect programming into a really practicable tool. These include:

- the number of different numerical codes needed to describe a) the model layout is kept very small, the more numerous set of codes required to control the finite-difference computation being produced automatically by a pre-developed program; the user's choice of codes to define the layout is checked
- b) graphically for errors;
- the finite-difference scheme employed can be changed by c) replacing one subroutine by another.

References

- Abbott, M.B., A. Damsgaard and G.S. Rodenhuis (1973), System 21, "Jupiter" (A Design System for Two-Dimensional Nearly Horizontal Flows), J. Hyd. Res., Vol 11, No.1, pp 1-28.
- Heaps, N.S. (1969). A Two-Dimensional Numerical Sea Model, Phil. Trans. Roy. Soc., London, Ser. A, Vol. 265, pp. 93-137.
- 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.
- Henry, R.F. (1982) Automated Programming of Explicit Shallow-Water Models. Part 1. Linearized Models with Linear or Quadratic Friction, Institute of Ocean Sciences, Sidney, B.C., Canada. Can. Tech. Rept. Hydrogr. Ocean Sci., No.3, 70 pp.
- Leendertse, J.J. (1967). Aspects of a Computational Model for Long-Period Water-Wave Propagation. RAND Corp., Santa Monica. RM-5294-PR, 165 pp.
- Sielecki, A. (1968). An Energy-Conserving Difference Scheme for the Storm Surge Equations. Mon. Weather Rev., Vol. 96, pp. 150-156.