Computers & Geosciences, Vol. 5, pp. 19–39
© Pergamon Press Ltd., 1979. Printed in Great Britain

0098-3004/79/0501-0019/$02.00/0

ADIF, A FORTRAN IV PROGRAM FOR SOLVING THE SHALLOW-WATER EQUATIONS

I. M. NAVON
National Research Institute for Mathematical Sciences, P.O. Box 395, Pretoria, South Africa

(*Received* 5 *May* 1978; *revised* 25 *October* 1978)

Abstract—A computer program is documented, implementing a linear Alternating Direction Implicit (ADI) method for a limited-area finite difference integration of the shallow-water equations on a β plane. Arbitrarily large time-steps can be used with this method, which is stable unconditionally for the linearized equations. The method also is efficient computationally, as the difference equations are factored into one-dimensional operators which approximately reproduce the original set of equations, and this obviates the necessity for solving a matrix of large bandwidth. A line-printer plot contouring the height field is generated as part of the output. Program options include the determination at each time step of two of the integral invariants of the shallow-water equation. For long-term runs either a dissipative term is provided, or a nonlinear lateral eddy viscosity coefficient of a friction force.

Key Words: Algorithm, FORTRAN, ADI, Shallow-water equations, Geophysics.

INTRODUCTION

The shallow-water equations system accurately describes wave motion on the surface of a homogeneous fluid when the horizontal wavelength is longer than both the vertical scale of motion and the depth of the fluid. As such the system is used widely in oceanography as a mathematical model for simulating circulation in coastal regions, lakes, etc. (Connor and Brebbia, 1976; Abbott, Donesgaard and Rodenhuis, 1973; Cheng, 1972; Connor and Wang, 1974).

In meteorological applications the same system of equations usually is termed 'the primitive barotropic equations' and has been used widely by numerous investigators for testing new numerical procedures prior to applying them to general three-dimensional models of the atmosphere (e.g. Grammeltvedt, 1969; Gustafsson, 1971; Cullen, 1973, 1974; Chen and Miyakoda, 1974; Merilees, 1973; Elvius and Sundström, 1973; Fairweather and Navon, 1977; to cite but a few).

As the shallow-water equations constitute a quasilinear system of hyperbolic equations, they are subjected to the Courant, Friedrichs, and Levy (CFL) stability condition when discretized by explicit time-difference approximations. However, in oceanographic and meteorological applications the time discretization error is small compared with the discretization error in space, and the short time-step imposed by the CFL stability condition can be avoided by using implicit time-differencing procedures.

Gustafsson (1971) was the first to propose an efficient fully implicit differencing method based on alternating direction techniques for solving the shallow-water equations, but his method requires systems of nonlinear algebraic equations to be solved at each time-step.

Fairweather and Navon (1977) proposed a linear ADI method for solving the shallow-water equations, the method being based on a perturbation of a linearized Crank–Nicolson type discretization, and requiring at each time-step only the solution of systems of linear algebraic equations. The method was determined to be more efficient computationally than the ADI method proposed by Gustafsson (1971).

A review of the linear ADI algorithm applied to the shallow-water equations on a β-plane for a test problem is presented in the next section, whereas the remainder of the report is devoted to the description of the program ADIF and specifications for its use. A listing of the FORTRAN IV source code of the program ADIF is included in Appendix A. Two variants of ADIF are presented, corresponding to the use of either the dissipative term or the nonlinear lateral eddy viscosity term for long-term runs. Typical outputs for 48-hour forecasts are presented in Appendix B, which includes a printer-plotted map of the height field.

REVIEW OF THE LINEAR ADI METHOD

The shallow-water equations

The shallow-water equations can be written (Houghton, Kasahara, and Washington, 1966)

$$\frac{\partial w}{\partial t} = A(w)\frac{\partial w}{\partial t} + B(w)\frac{\partial w}{\partial y} + C(y)w$$
$$0 \le x \le L, \quad 0 \le y \le D, \quad t \ge 0. \tag{1}$$

where L and D are the dimensions of a rectangular domain of area $\bar{A} = L \cdot D$. w is the vector function,

$$w = (u, v, \Phi)^T, \tag{2}$$

u, v are the velocity components in the x and y directions respectively, and

$$\Phi = 2\sqrt{gh} \tag{3}$$

where h is the depth of the fluid and g is the acceleration of gravity.

In (1) A, B, and C are matrices given by

$$A = -\begin{bmatrix} u & 0 & \Phi/2 \\ 0 & u & 0 \\ \Phi/2 & 0 & u \end{bmatrix} \quad B = -\begin{bmatrix} v & 0 & 0 \\ 0 & v & \Phi/2 \\ 0 & \Phi/2 & v \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & f & 0 \\ -f & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{4}$$

Where f is the Coriolis term given by

$$f = \bar{f} + \beta(y - D/2) \tag{5}$$

with $\bar{f}$ and β constants.

Periodic boundary conditions are assumed in the x direction:

$$w(x, y, t) = w(x + L, y, t), \tag{6}$$

whereas in the y-direction the boundary condition is

$$v(x, 0, t) = v(x, D, t) = 0. \tag{7}$$

With those boundary conditions and with the initial condition $w(x, y, 0) = \varphi(x, y)$ the total energy

$$E = \frac{1}{2}\int_0^L \int_0^D (u^2 + v^2 + \Phi^2/4)\Phi^2/4g \, dx \, dy \tag{8}$$

is independent of time.

Also, the average value of the height of the free surface is conserved, that is,

$$\bar{h} = \frac{\int_0^L \int_0^D h \, dx \, dy}{\bar{A}} \tag{9}$$

in independent of time.

The linear ADI algorithm

Let N_x and N_y be positive integers and set

$$\Delta x = L/N_x, \quad \Delta y = D/N_y. \tag{10}$$

We shall denote by w^n_{jk} an approximation to $w(j\Delta x, k\Delta y, n\Delta t)$ where Δt is the time-step. The basic difference operators are

$$\begin{aligned} D_{0x} w^n_{jk} &= (w^n_{j+1,k} - w^n_{j-1,k})/(2\Delta x), \\ D_{+x} w^n_{jk} &= (w^n_{j+1,k} - w_j{}^n k)/\Delta x, \\ D_{-x} w^n_{jk} &= (w^n_{jk} - w^n_{j-1,k})/\Delta x \end{aligned} \tag{11}$$

respectively, with similar definitions for D_{0y}, D_{+y}, and D_{-y}. We also define the operators P^n_{jk} and Q^n_{jk} by

$$\begin{aligned} P^n_{jk} &= \frac{\Delta t}{2}(A(w^n_{jk})D_{0x} + C_k^{(1)}, \\ Q^n_{jk} &= \frac{\Delta t}{2}(B(w^n_{jk})D_k + C_k^{(2)}, \end{aligned} \tag{12}$$

where

$$C_k^{(1)} = \begin{bmatrix} 0 & 0 & 0 \\ -f_k & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad C_k^{(2)} = \begin{bmatrix} 0 & f_k & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{13}$$

and where, due to boundary conditions in the y direction,

$$D_k = \begin{cases} D_{0y} & \text{for} \quad k = 1, 2, \ldots, N_{y-1}, \\ D_{+y} & \text{and} \quad k = 0, \\ D_{-y} & \text{for} \quad k = N_y. \end{cases} \tag{14}$$

The linear ADI algorithm for the shallow-water equations (Fairweather and Navon, 1977) is given by

$$(I - P^{n*}_{jk})w^{(n+1)*}_{jk} = V^n_{jk} \tag{15a}$$

$$(I - Q^{n*}_{jk})w^{n+1}_{jk} = 2w^{(n+1)*}_{jk} - V^n_{jk} \tag{15b}$$

with

$$V^n_{jk} = (I + Q^{n*}_{jk})w^n_{jk} \tag{16}$$

and where

$$\begin{aligned} P^{n*}_{jk} &= \frac{\Delta t}{2}(A(\hat{w}^n_{jk})D_{0x} + C_k^{(1)} \\ Q^{n*}_{jk} &= \frac{\Delta t}{2}(B(\hat{w}^n_{jk})D_k + C_k^{(2)} \end{aligned} \tag{17}$$

$$\hat{w}^n_{jk} = 1/2(3w^n_{jk} - w^{n-1}_{jk}), \quad n \geq 1 \tag{18a}$$

$$\hat{w}^0_{jk} = w^0_{jk} + (P^{(0)}_{jk} + Q^{(0)}_{jk})w^0_{jk}, \tag{18b}$$

and where $w^{(n+1)*}_{jk}$ is an auxiliary solution. Note that owing to (18) we now have to solve only sequences of linear systems of algebraic equations. Also, owing to the assumption of periodic boundary conditions in the x direction, the coefficient matrices which arise where the linear ADI algorithm is applied along horizontal rows, are either block or scalar cyclic tridiagonal.

PROGRAM ADIF

The test problem

The initial height field condition No. 1 of Grammeltvedt (1969) was used as test problem:

$$h(x, y) = H_0 + H_1 \tanh\left(\frac{9(D/2 - y)}{2D}\right) + H_2 \operatorname{sech}^2\left(\frac{9(D/2 - y)}{D}\right) \cdot \sin\left(\frac{2\pi x}{L}\right). \tag{19}$$

The initial velocity fields were derived from the initial height field using the geostrophic relationship

$$u = \left(\frac{-g}{f}\right)\frac{\partial h}{\partial y}, \quad v = \left(\frac{g}{f}\right)\frac{\partial h}{\partial x}. \qquad (20$$

The constants used were

$$\begin{aligned} &L = 4400 \text{ km} && g = 10 \text{ m sec}^{-2} \\ &D = 6000 \text{ km} && H_0 = 2000 \text{ m} \\ &\bar{f} = 10^{-4} \text{ sec}^{-1} && H_1 = 220 \text{ m} \\ &\beta = 1.5 \cdot 10^{-11} \text{ sec}^{-1} \text{ m}^{-1} && H_2 = 133 \text{ m}. \end{aligned} \tag{21}$$

The time and space increments used for short runs (2 days) were

$$\begin{aligned}&\text{(a)}\quad \Delta x = \Delta y = 200\,\text{km} \quad \Delta t = 1800\,\text{sec}\\ &\text{(b)}\quad \Delta x = \Delta y = 200\,\text{km} \quad \Delta t = 3600\,\text{sec.}\end{aligned} \tag{22}$$

For long-term integrations (20 days) the time and space increments were

$$\Delta x = \Delta y = 500\,\text{km} \quad \Delta t = 3600\,\text{sec.} \tag{23}$$

Dissipation term

To avoid nonlinear instabilities in long-term integrations a dissipation term of the form

$$\epsilon \Delta t^3 [D_{+x} D_{-x} + D_{+y} D_{-y}] w_{jk}^n \tag{24}$$

was added to the right-hand side of equation (15a) rendering our approximation dissipative of order 2 (Kreiss and Oliger, 1973). An optimal value of $\epsilon = 0.020$ was determined experimentally (Fairweather and Navon, 1977) and it permitted the conservation of total energy and mean height after 20 days of numerical integration.

Nonlinear eddy diffusion

Another method of handling nonlinear instabilities, and one which is employed frequently in modeling ocean circulations, is to use a nonlinear eddy viscosity term. In the program ADIF, by calling subroutine EDDY (described here), an option is included which includes a nonlinear eddy viscosity term. The calculation is based upon two-dimensional turbulence theory.

Following Haney and Wright (1975) the eddy viscosity coefficient is written as

$$\nu = \nu_0 (1 + \gamma |\nabla \xi| (\Delta x^3)), \tag{25}$$

where ν_0 and γ are constants, ξ is the relative vorticity and ∇ is the horizontal del-operator.

To equations (15a) and (15b) are added the terms F_x and F_y, the x and y components respectively of the friction force due to lateral eddy viscosity, and given by

$$F_x = \nabla \cdot (\nu \nabla u), \quad F_y = \nabla \cdot (\nu \nabla v). \tag{26}$$

In subroutine EDDY the same finite-difference approximation is followed as that of Haney and Wright (1975).

Another expression which also can be used in subroutine EDDY for the nonlinear eddy viscosity coefficient was given by Crowley (1970):

$$\nu = \bar{a}^{3/2} |\nabla \xi| \nabla x^3, \tag{27}$$

where $\bar{a}$ is a dimensionless coefficient.

Fairweather and Navon (1977) determined that when nonlinear instabilities are being dealt with, the second-order dissipation term is more efficient.

Program operations

Explanatory comments are included in the program listing in Appendix A. We shall first describe the input specifications and then the various subroutines of program ADIF.

(i) *Input specifications.* The input to the program consists of two cards, as follows.

Card 1: FORMAT (6E10.4,2I5,F10.0) contains the following nine parameters.

FL—the length dimension (L) of the rectangular domain;
D—the width dimension (D) of the rectangular domain;
T—total simulation time (in seconds);
DX—the space increment in the x direction in meters;
DY—the space increment in the y direction in meters;
DT—the time-step in seconds;
IPR—a parameter controlling output operations of the program, that is specifying after how many time steps the forecast fields should be displayed;
MM—a parameter specifying which version of the ADIF program was used;
ADJ—the value of the coefficient of the diffusive term (ϵ).

Card 2 (Called in subroutine SETUP) specifies different parameters relative to the initial field [see equation (19)] using format 6E10.4, and contains the following five parameters.

$H0$—constant for the initial height field;
$H1$—constant for the initial height field;
$H2$—constant for the initial height field;
FHAT—Coriolis parameter;
BETA—df/dy, the Rossby parameter.

(ii) *Main program and subroutines.* The main program SHALLOW reads the first data card and after some preliminary calculations calls the subroutines SETUP, LOOK, WSTAR, and UVOUT which perform the initial fields calculations and display the initial fields. After calculating the diffusivity term (or the nonlinear eddy-diffusion term by calling subroutine EDDY) the program loops each time step on the central subroutine ADI which performs the bulk of the linear ADI algorithm calculation and in turn calls the subroutines CYCBLK and BLKTRID, CYCTRID, and TRIDIAG. These subroutines solve the scalar and block-cyclic tridiagonal systems of linear algebraic equations resulting from the ADI-discretization of the shallow-water equations.

After a predetermined number of time-steps subroutine LOOK is called, which calculates the integral invariants of total energy and mean height and in turn calls subroutines HOUT and MAPPA. These subroutines perform the printing and the line-printer plotting of the height field respectively.

Finally, after the preset simulation time has been reached, the u and v velocity field components are printed by calling subroutine UVOUT, and the final height and velocity fields written on disk for further use.

Subroutine SETUP (U, V, PHI, H, NX, NY, S, C, *LX*). This subroutine sets up the initial height field *H*, calculates the variable PHI $= \Phi = 2\sqrt{gh}$ and from it, using equation (20), the components of the initial velocity field *U* and *V*; the subroutine also calculates the Coriolis parameter *F*. The parameters *NX*, *NY*, and *LX* are calculated in the main program SHALLOW to be the effective number of space increments in the x and y directions respectively, whereas *LX* is the maximal number of space increments in the x direction.

S and *C* are auxiliary parameters for calculating intermediate trigonometric variables.

Subroutine LOOK (*U*, *V*, *PHI*, *H*, *NX*, *NY*, *LX*). This subroutine calculates at each time step the total energy and mean height which are invariants of the shallow-water equations. It also prints these values, together with the height–field values, by calling subroutine HOUT, and calls subroutine MAPPA for a line-printer contour plot of the height field. The CPU time for each 12 time steps is also printed.

Subroutine MAPPA (*FUN*, *C*, *NX*, *NZ*, *LX*). This subroutine provides a visual display of the height field by line-printing an isoline contour plot of the height field for every 50 meters. The parameter FUN gives the forecast field to be contoured, whereas the parameter *C* is the inverse of the contour distance in meters (e.g. if we wish a 50-m contour distance, $C = 0.02$). The parameter $NZ = NY + 1$.

Subroutine HOUT (*H*, *NX*, *NY*, *LX*). This subroutine digitally prints the height field values in a matrix format.

Subroutine WSTAR (*U*, *V*, *PHI*, *US*, *VS*, *PHIS*, *F*, *NX*, *NY*, *LX*). This subroutine calculates the starting values for the application of the ADI algorithm following equation (18b). It also imposes the corresponding boundary conditions. *U*, *V*, and *PHI* stand for the initial fields whereas *US*, *VS*, *PHIS* are the calculated fields following equation (18b). *F* is the Coriolis parameter.

Subroutine UVOUT (*W*, *NX*, *NY*, *LX*). This subroutine digitally prints the values of the velocity field components in a matrix format. *W* stands for either the *U* or the *V* component of the velocity field.

Subroutine ADI (*U*, *V*, *PHI*, *US*, *VS*, *PHIS*, *UBS*, *VBS*, *PHIBS*, *F*, *A*, *B*, *C*, *E*, *W*, *NX*, *NY*, *LX*, *UN*, *VN*, *PHIN*, *P*, *Q*, *R*, *S*). This subroutine performs the bulk of the work in the use of the linear ADI algorithm to solve the shallow-water equations.

The variable *IND* assumes the values 0, 1, or 2. The value 0 corresponds to the situation when the dissipative term given by equation (24) is used. The values 1 and 2 are assumed if a nonlinear eddy diffusion term, given by equations (25) or (27) correspondingly, is included in the model.

The arrays *A*, *B*, and *C* describe the entries of the individual 2×2 block matrices belonging to a series of $N_y + 1$ block-cyclic tridiagonal systems obtained from the application of the linear ADI algorithm to the coupled variables (u, Φ) following equation (15a). The matrix *E* describes the right-hand side of the block-cyclic tridiagonal systems. The solution of those systems is obtained by calling subroutine CYCBLK which in turn calls subroutine BLKTRID.

Next a series of $N_y + 1$ scalar cyclic tridiagonal systems have to be solved for the v field. The vectors **P**, **W**, **R** describe the entries of those vectors whereas the vector **S** describes the right-hand sides of the scalar cyclic tridiagonal systems. The solution of those systems is obtained by calling subroutine CYCTRID which in turn calls subroutine TRIDIAG.

The intermediate fields thus obtained are renamed *VBS*, *PHIBS*, and *VBS* and correspond to $w_{jk}^{(n+1)*}$ of equation (15a).

In the second part of the ADI algorithm we first solve for the coupled variables (v, Φ). The coefficient matrices now are block-tridiagonal and are solved by calling the block-tridiagonal solver, subroutine BLKTRID. The arrays *A*, *B*, *C*, and *E* are overwritten. Then the variable *U* is determined by solving a series of tridiagonal coefficient matrices by calling subroutine TRIDIAG.

The new field variables thus obtained u_{jk}^{n+1}, v_{jk}^{n+1}, ϕ_{jk}^{n+1} are renamed *UN*, *VN*, and *PHIN* respectively.

Next, new starting values *US*, *VS*, *PHIS* are determined using equation (18a), whereas *UN*, *VN*, *PHIN* now become the current values for a new linear ADI time-step. For a detailed description of the linear ADI algorithm and the algebraic procedure (see Fairweather and Navon, 1977).

Subroutine CYCBLK (*N*, *P*, *Q*, *R*, *D*, *W*). This is a cyclic block-tridiagonal solver written so as to take full advantage of the fact that the blocks in the coefficient matrices are 2×2. The algorithm used (see Navon, 1977) is a generalization to block-cyclic tridiagonal systems of the algorithm given by Ahlberg, Nilson, and Walsh in 1967 (the ANW-algorithm).

The definitions of the arguments are as follows:

P, *W*, and *R* are arrays of dimension $4 * \mathrm{N}$ which contain the elements of the 2×2 blocks along the subdiagonal, diagonal, and superdiagonal respectively;

D is an array dimensioned $2 * N$ containing the right-hand side;

W is a working-area array of dimension not less than $2 * N * 3$.

The solution vectors are stored in array *D*. $N = NX$. During the computation all input arrays are overwritten.

Subroutine BLKTRID (*N*, *M*, *A*, *B*, *C*, *D*, *L*). This subroutine is a specific block-tridiagonal solver written for the situation when the individual blocks are 2×2. In the subroutine use is made of a direct-solution method based on block-Gaussian elimination without pivoting. The parameters *A*, *B*, *C*, *D* represent the same arrays as *P*, *Q*, *R*, *D* of subroutine CYCBLK.

M takes the value 3 when the subroutine BLKTRID is called from CYCBLK and the value 1 when the subroutine BLKTRID is called from subroutine ADI, in the second part of the linear ADI algorithm. *L* stands for either *NX* or *NY* depending on whether we are solving equation (15a) or equation (15b).

Subroutine CYCTRID (*N*, *P*, *Q*, *R*, *D*, *W*). This subroutine solves a scalar cyclic tridiagonal system using the ANW algorithm and is called only once for a given matrix. The definitions of the arguments are as follows.

P, *Q*, and *R* are *N*-element vectors containing subdiagonal, diagonal, and superdiagonal elements. $N = NX$.

W is a working-area array of dimension $2 * N$.

D is the right-hand side *N*-element vector. The solution vector finally is overwritten on *D*.

Subroutine TRIDIAG (*N, M, A, B, C, D, L*). This is a classical tridiagonal solver. The definitions of the arguments are as follows.

N is either *NX*-1, when TRIDIAG is called from subroutine CYCTRID, or *NY*, when TRIDIAG is called from subroutine ADI.

M is either 2, when TRIDIAG is called from subroutine CYCTRID, or 1, when TRIDIAG is called from subroutine ADI.

A, B, and *C* have the same meaning as the vectors **P**, **Q**, **R** in subroutine CYCTRID.

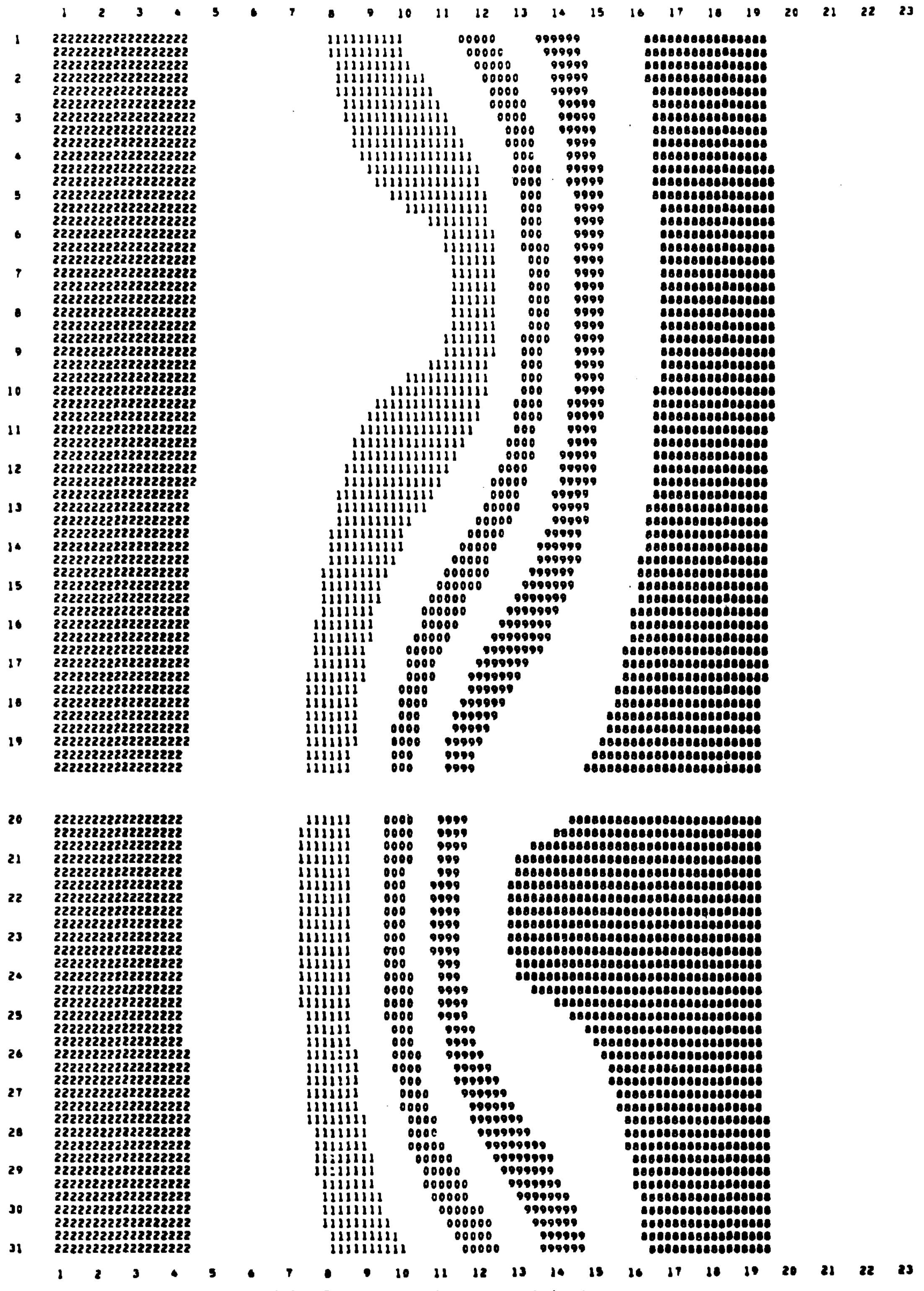

Figure 1. Initial height field contours with space resolution $\Delta x = \Delta y = 200$ km; CI = 50 m.

D stands for the *N*-component vector S when TRIDIAG is called from subroutine ADI, or it stands for the working-area array *W*, dimensioned $2 * N$, when TRIDIAG is called from subroutine CYCTRID.

L stands for either *NX* or *NY*.

Subroutine EDDY (*U*, *V*, *EVX*, *EVY*, *LX*, *NX*, *NY*). This subroutine calculates the x and y components of the friction force due to nonlinear lateral eddy viscosity, the components being here denoted *EVX* and *EVY* respectively. The nonlinear eddy viscosity coefficient can be calculated following equation (25) or equation (27) by setting an index *IND* equal to 1 or 2 respectively.

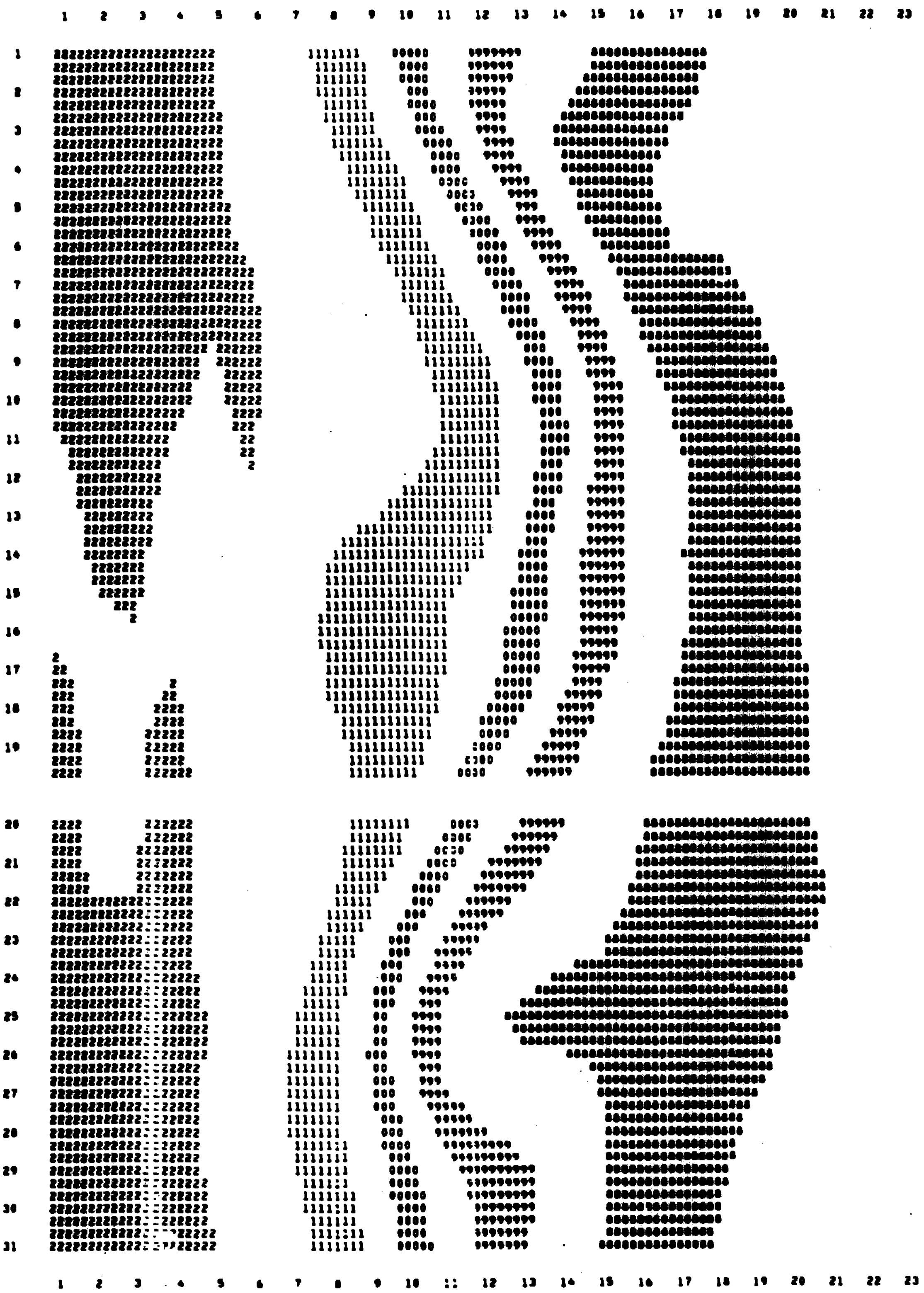

Figure 2. Height field contours after 48 hr. Space resolution $\Delta x = \Delta y = 200$ km; CI = 50 m; $\Delta t = 1800$ sec.

APPENDIX A—LISTING IN FORTRAN IV OF SOURCE CODE FOR ADIF

```
      PROGRAM SHALLOW(INPUT,OUTPUT,TAPE1=INPUT,TAPE3=OUTPUT,TAPE4)
      COMMON/CONST/FL,D,T,DX,DY,DT,FX,FY,FT,G,TIME,IPR,ADJ,BDJ
      COMMON/EDVORT/GNU,GAM,GDX3,PX,PY,PXSQ,PYSQ,PXS2,PYS2,ALPHA,FAC,IND
      DIMENSION U(35,25),V(35,25),PHI(35,25),US(35,25),VS(35,25),
     1 PHIS(35,25),UBS(35,25),VBS(35,25),PHIBS(35,25),UN(35,25),
     2 VN(35,25),PHIN(35,25),H(35,25),F(35),A(4,35),B(4,35),C(4,35),
     3 E(2,35),P(35),Q(35),R(35),S(35),W(2,35,3)
      EQUIVALENCE (UBS(1),UN(1),H(1)),(VBS(1),VN(1)),(PHIBS(1),PHIN(1))
      DATA NAME/"FW4"/
      DATA G/10.0/
      DATA ITAPE1/1/
      DATA ITAPE4/4/
      DATA ITAPE3/3/
    1 FORMAT(6E10.4,2I5,F10.0)
    2 FORMAT("0  THIS RUN WITH VERSION ",A3,I2)
    3 FORMAT("0                ADJ=",F11.8)
    4 FORMAT(" NU = ",E10.3,5X,"GAMMA = ",E10.3)
    5 FORMAT("0 ALPHA = ",E12.4)
    7 FORMAT("0 CHANGE DIMENSIONS OF ARRAYS  U,V,......,W  TO ACCOMMODATE
     1 THIS DATA SET,"/5X,"AND THE VALUES ASSIGNED TO  LX  AND  LY, WHIC
     2H INDICATE CERTAIN ARRAY DIMENSIONS.")
    8 FORMAT("1     INITIAL U-FIELD")
    9 FORMAT("1     INITIAL V-FIELD")
      GNU=0.
      GAM=0.
      ALPHA=0.
      READ(ITAPE1,1)FL,D,T,DX,DY,DT,IPR,MM,ADJ
      LX=35
      LY=25
      NX=FL/DX
      NY=1+IFIX(D/DY)
      IF(NX.GT.LX) GO TO 45
      IF(NY.LE.LY) GO TO 50
   45 WRITE(ITAPE3,7)
      GO TO 250
   50 NT=T/DT
      IF(IPR.EQ.0) IPR=1
      FX=DT/(4.*DX)
      FY=DT/(4.*DY)
      FT=0.5*DT
      CALL SETUP(U,V,PHI,H,F,NX,NY,A,B,LX)
      WRITE(ITAPE3,3) ADJ
      WRITE(ITAPE3,4) GNU,GAM
      WRITE(ITAPE3,5) ALPHA
      TIME=0.
      CALL LOOK(U,V,PHI,H,F,NX,NY,LX)
      CALL WSTAR(U,V,PHI,US,VS,PHIS,F,NX,NY,LX)
      WRITE(ITAPE3,8)
      CALL UVOUT(U,NX,NY,LX)
      WRITE(ITAPE3,9)
      CALL UVOUT(V,NX,NY,LX)
      IOPT=0
      IND=0
      IF(ALPHA.EQ.0.) GO TO 90
      IND=2
      GO TO 100
   90 IF(GNU.EQ.0..AND.GAM.EQ.0.) GO TO 95
      IND=1
      GO TO 100
   95 IF(ADJ.EQ.0.) GO TO 100
      EPSDT=ADJ*DT*DT*DT
      ADJ=EPSDT/(DY*DY)
      BDJ=EPSDT/(DX*DX)
  100 CONTINUE
C   LOOP FOR EACH TIME STEP
      DO 200 I=1,NT
```

3 show the height field contours after two days of simulation using time steps of 1800 and 3600 sec respectively. The contour interval is 50 m and the digits 8, 9, 0, 1, 2 stand for 1800, 1900, 2000, 2100, 2200 meters respectively. Figures 4 and 5 show the height field contours for a long-term integration using a space resolution of $\Delta x = \Delta y = 500$ km and a time step of 3600 sec, the dissipation coefficients being $\epsilon = 0.015$ and $\epsilon = 0.020$ respectively.

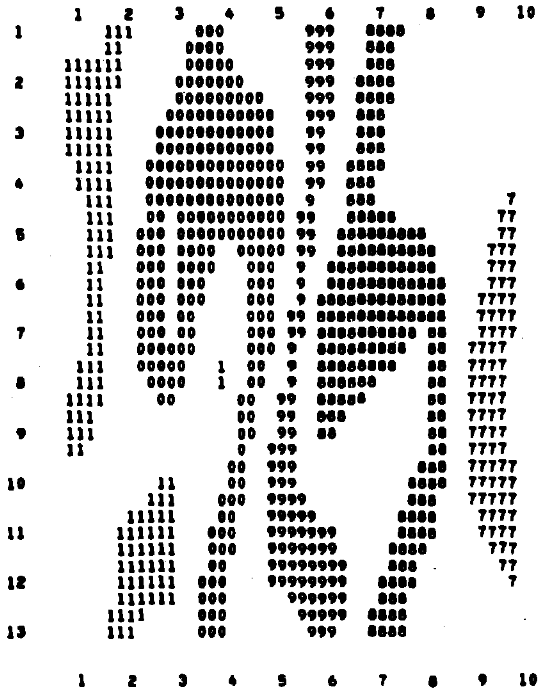
Figure 4. Height field contours after 20 days. Space resolution $\Delta x = \Delta y = 500$ km; CI = 50 m; $\Delta t = 1800$ sec; $\epsilon = 0.015$.

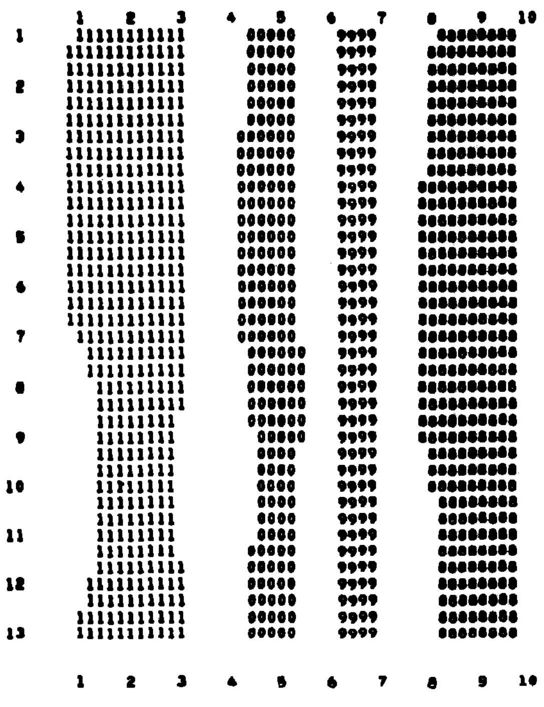
Figure 5. Height field contours after 20 days. Space resolution $\Delta x = \Delta y = 500$ km; CI = 50 m; $\Delta t = 3600$ sec; $\epsilon = 0.020$.

Acknowledgment—Thanks are due to Miss J. Hewitt of the National Research Institute for Mathematical Sciences of the Council for Scientific and Industrial Research, who did the programming.

REFERENCES

Abbott, M. B., Danesgaard, A., and Rodenhuis, G. S., 1973, System 21 Jupiter (a design system for two-dimensional nearly-horizontal flows): Jour. Hydraulic Research, v. 11, no. 1, p. 1–28.

Ahlberg, H. H., Nilson, E. N., and Walsh, J. L., 1967, The theory of splines and their applications, *in* Mathematics in science and engineering, v. 38: Academic Press, New York 289 p.

Chen, H. H., and Miyakoda, K., 1974, A nested grid computation for the barotropic free surface atmosphere: Mon. Wea. Rev., v. 102, p. 181–190.

Cheng, R. T., 1972, Numerical investigation of lake circulation around islands by the finite-element method: Intern. Jour. Numer. Math. in Eng., v. 5, no. 1, p. 103–112.

Connor, J. J., and Brebbia, C. A., 1976, Finite-element techniques for fluid flow: Newnes-Butterworths Ed., London, 310 p.

Connor, J. J., and Wang, J., 1974, Finite-element modelling of hydrodynamic circulation, *in* Numerical methods in fluid dynamics: Pentech Press, London, p. 355–387.

Cullen, M. J. P., 1973, A simple finite-element method for meteorological problems: Jour. Inst. Math. Appl., v. 11, p. 15–31.

Cullen, M. J. P., 1974, A finite-element method for a non-linear initial-value problem: Jour. Inst. Math. Appl., v. 13, p. 233–247.

Crowley, W. P., 1970, A numerical model for viscous, free-surface barotropic wind-driven ocean circulations: Jour. Comp. Phys., v. 5, p. 139–168.

Elvius, T., and Sundström, A., 1973, Computationally efficient schemes and boundary conditions for a fine-mesh barotropic model based on the shallow-water equations: Tellus, v. 25, no. 2, p. 132–156.

Fairweather, G., and Navon, I. M., 1977, A linear alternating-direction implicit (ADI) method for solving the shallow-water equations: Council for Scientific and Industrial Research (CSIR), Sp. Rept. WISK 269, Pretoria, South Africa, 20 p.

Grammeltvedt, A., 1969, A survey of finite-difference schemes for the primitive equations for a barotropic fluid: Mon. Wea. Rev., v. 97, no. 5, p. 384–404.

Gustafsson, B., 1971, An alternating-direction implicit method for solving the shallow-water equations: Jour. Comp. Phys., v. 7, p. 239–254.

Haney, R. L., and Wright, J. M. Jr., 1975, The relationship between the grid size and the coefficient of non-linear lateral eddy viscosity in numerical ocean circulation models: Jour. Comp. Phys., v. 19, p. 257–266.

Houghton, D., Kasahara, A., and Washington, W., 1966, Long-term integration of the barotropic equations by the Lax-Wendroff method: Mon. Wea. Rev., v. 94, p. 141–150.

Kreiss, H. O., and Oliger, J., 1973, Methods for the approximate solution of time-dependent problems: Global Atmospheric Research Programme (GARP) Publ. Ser. No. 10, World Met. Org., 107 p.

Merilees, P. E., 1973, The pseudo-spectral approximation applied to the shallow-water equations on a sphere: Atmosphere, v. 11, p. 13–20.

Navon, I. M., 1977, Algorithms for the solution of scalar and block-cyclic tridiagonal systems: Council for Scientific and Industrial Research (CSIR), Sp. Rept. WISK 265, Pretoria, South Africa, 29 p.

APPENDIX A—LISTING IN FORTRAN IV OF SOURCE CODE FOR ADIF

```
      PROGRAM SHALLOW(INPUT,OUTPUT,TAPE1=INPUT,TAPE3=OUTPUT,TAPE4)
      COMMON/CONST/FL,D,T,DX,DY,DT,FX,FY,FT,G,TIME,IPR,ADJ,BDJ
      COMMON/EDVORT/GNU,GAM,GDX3,PX,PY,PXSQ,PYSQ,PXS2,PYS2,ALPHA,FAC,IND
      DIMENSION U(35,25),V(35,25),PHI(35,25),US(35,25),VS(35,25),
     1 PHIS(35,25),UBS(35,25),VBS(35,25),PHIBS(35,25),UN(35,25),
     2 VN(35,25),PHIN(35,25),H(35,25),F(35),A(4,35),B(4,35),C(4,35),
     3 E(2,35),P(35),Q(35),R(35),S(35),W(2,35,3)
      EQUIVALENCE (UBS(1),UN(1),H(1)),(VBS(1),VN(1)),(PHIBS(1),PHIN(1))
      DATA NAME/"FW4"/
      DATA G/10.0/
      DATA ITAPE1/1/
      DATA ITAPE4/4/
      DATA ITAPE3/3/
    1 FORMAT(6E10.4,2I5,F10.0)
    2 FORMAT("0   THIS RUN WITH VERSION ",A3,I2)
    3 FORMAT("0                   ADJ=",F11.8)
    4 FORMAT(" NU = ",E10.3,5X,"GAMMA = ",E10.3)
    5 FORMAT("0 ALPHA = ",E12.4)
    7 FORMAT("0 CHANGE DIMENSIONS OF ARRAYS  U,V,......,W  TO ACCOMMODATE
     1 THIS DATA SET,"/5X,"AND THE VALUES ASSIGNED TO  LX  AND  LY, WHIC
     2H INDICATE CERTAIN ARRAY DIMENSIONS.")
    8 FORMAT("1     INITIAL U-FIELD")
    9 FORMAT("1     INITIAL V-FIELD")
      GNU=0.
      GAM=0.
      ALPHA=0.
      READ(ITAPE1,1)FL,D,T,DX,DY,DT,IPR,MM,ADJ
      LX=35
      LY=25
      NX=FL/DX
      NY=1+IFIX(D/DY)
      IF(NX.GT.LX) GO TO 45
      IF(NY.LE.LY) GO TO 50
   45 WRITE(ITAPE3,7)
      GO TO 250
   50 NT=T/DT
      IF(IPR.EQ.0) IPR=1
      FX=DT/(4.*DX)
      FY=DT/(4.*DY)
      FT=0.5*DT
      CALL SETUP(U,V,PHI,H,F,NX,NY,A,B,LX)
      WRITE(ITAPE3,3) ADJ
      WRITE(ITAPE3,4) GNU,GAM
      WRITE(ITAPE3,5) ALPHA
      TIME=0.
      CALL LOOK(U,V,PHI,H,F,NX,NY,LX)
      CALL WSTAR(U,V,PHI,US,VS,PHIS,F,NX,NY,LX)
      WRITE(ITAPE3,8)
      CALL UVOUT(U,NX,NY,LX)
      WRITE(ITAPE3,9)
      CALL UVOUT(V,NX,NY,LX)
      IOPT=0
      IND=0
      IF(ALPHA.EQ.0.) GO TO 90
      IND=2
      GO TO 100
   90 IF(GNU.EQ.0..AND.GAM.EQ.0.) GO TO 95
      IND=1
      GO TO 100
   95 IF(ADJ.EQ.0.) GO TO 100
      EPSDT=ADJ*DT*DT*DT
      ADJ=EPSDT/(DY*DY)
      BDJ=EPSDT/(DX*DX)
  100 CONTINUE
C   LOOP FOR EACH TIME STEP
      DO 200 I=1,NT
      IOPT=IOPT+1
  251 CALL ADI(U,V,PHI,US,VS,PHIS,UBS,VBS,PHIBS,F,A,B,C,E,W,NX,NY,LX,
     1 UN,VN,PHIN,P,Q,R,S)
  252 TIME=TIME+DT
      IF(IOPT.GE.IPR) GO TO 195
      IF(I.LT.NT) GO TO 200
C     PRINT VALUE OF ENERGY INTEGRAL AT THIS TIME STEP.
C     PRINT HEIGHT VALUES AT THIS TIME STEP.
  195 CALL LOOK(U,V,PHI,H,F,NX,NY,LX)
      IOPT=0
  200 CONTINUE
C  END OF TIME STEP LOOP
      CALL UVOUT(U,NX,NY,LX)
      CALL UVOUT(V,NX,NY,LX)
      DAYS=T/86400.
      NINT=1
      REWIND ITAPE4
      WRITE(ITAPE3,2) NAME,MM
      WRITE(ITAPE4) NAME,NINT,MM,DAYS,DT,DX,DY,NX,NY
      WRITE(ITAPE4) ((U(J,K),J=1,NX),K=1,NY)
      WRITE(ITAPE4) ((V(J,K),J=1,NX),K=1,NY)
      WRITE(ITAPE4) ((PHI(J,K),J=1,NX),K=1,NY)
      REWIND ITAPE4
  250 STOP
      END
      SUBROUTINE WSTAR (U,V,PHI,US,VS,PHIS,F,NX,NY,LX)
C  TO SET UP INITIAL VALUES OF US, VS, PHIS  FOR ALL J,K
      COMMON/CONST/FL,D,T,DX,DY,DT,FX,FY,FT,G,TIME,IPR,ADJ,BDJ
C          0*           0        0        0           0                 T
C   OMEGA     = OMEGA     +( P     + Q      ) OMEGA      ,  OMEGA=(U,V,PHI)
C        J,K          J,K      J,K      J,K          J,K
      DIMENSION U(LX,NY),V(LX,NY),PHI(LX,NY),US(LX,NY),VS(LX,NY),
     1 PHIS(LX,NY),F(NY)
      DATA ITAPE3/3/
      NYM=NY-1
      DO 20 K=1,NY
      FK=F(K)
      QY=FY
      KP1=K+1
      KM1=K-1
      IF(KP1.LE.NY) GO TO 10
      KP1=NY
      GO TO 11
   10 IF(KM1.GE.1) GO TO 12
      KM1=1
   11 QY=FY+FY
   12 DO 15 J=1,NX
      JP1=J+1
      IF(JP1.GT.NX) JP1=1
      JM1=J-1
      IF(JM1.LT.1) JM1=NX
      UKD=U(JP1,K)-U(JM1,K)
      PKD=PHI(JP1,K)-PHI(JM1,K)
      VJD=V(J,KP1)-V(J,KM1)
      PJD=PHI(J,KP1)-PHI(J,KM1)
      PHI2=PHI(J,K)/2.
      US(J,K)=U(J,K)-FX*(U(J,K)*UKD+PHI2*PKD)-V(J,K)*(QY*(U(J,KP1)
     1 -U(J,KM1))-FK)
      VS(J,K)=V(J,K)-U(J,K)*(FX*(V(JP1,K)-V(JM1,K))+FK)
     1 -QY*(V(J,K)*VJD+PHI2*PJD)
      PHIS(J,K)=0.5*(PHI(J,K)-FX*(PHI2*UKD+U(J,K)*PKD)-QY*(PHI2*VJD
     1 +V(J,K)*PJD))
   15 CONTINUE
   20 CONTINUE
      DO 25 J=1,NX
      VS(J,1)=0.
   25 VS(J,NY)=0.
      RETURN
      END
      SUBROUTINE ADI(U,V,PHI,US,VS,PHIS,UBS,VBS,PHIBS,F,A,B,C,E,W,NX,NY,
     1 LX,UN,VN,PHIN,P,Q,R,S)
      COMMON/CONST/FL,D,T,DX,DY,DT,FX,FY,FT,G,TIME,IPR,ADJ,BDJ
      COMMON/EDVCRT/GNU,GAM,GDX3,PX,PY,PXSQ,PYSQ,PXS2,PYS2,ALPHA,FAC,IND
      DIMENSION U(LX,NY),V(LX,NY),PHI(LX,NY),US(LX,NY),VS(LX,NY),
     1 PHIS(LX,NY),UBS(LX,NY),VBS(LX,NY),PHIBS(LX,NY),F(NY),A(4,NY),
     2 B(4,NY),C(4,NY),E(2,NX),S(NX),     W(2,NX,3),UN(LX,NY),VN(LX,NY),
     3 PHIN(LX,NY),P(NY),Q(NY),R(NY)
      DIMENSION EVX(35,25),EVY(35,25)
      IF(IND.EQ.0) GO TO 25
      IF(IND.EQ.1) GDX3=GAM*DX*DX*DX
      IF(IND.EQ.2) FAC=DX*DX*DX*ALPHA*SQRT(ALPHA)
      IF(TIME.GT.0.) GO TO 20
      PX=0.5/DX
      PY=0.5/DY
      PXSQ=PX*PX
      PYSQ=PY*PY
      PXS2=PXSQ+PXSQ
      PYS2=PYSQ+PYSQ
   20 CALL EDDY(U,V,EVX,EVY,LX,NX,NY)
C     LOOP FOR EACH HORIZONTAL LINE  Y(K), K=1,...,NY
   25 DO 140 K=1,NY
      FK=F(K)
      FU=FY
      KP1=K+1
      KM1=K-1
      IF(KP1.LE.NY) GO TO 105
      KP1=NY
      GO TO 110
  105 IF(KM1.GE.1) GO TO 115
      KM1=1
  110 FU=FU+FU
C      LOOP FOR EACH POINT  X(J), J=1,...,NX  ON LINE  Y(K).
  115 DO 120 J=1,NX
C         SET UP ELEMENTS OF CYCLIC BLOCK (2*2) TRIDIAGONAL MATRIX, AND
C              CONSTANT TERMS FOR DETERMINATION OF  UBS AND PHIBS.
      JP1=J+1
      JM1=J-1
      IF(JM1.LT.1) JM1=NX
      IF(JP1.GT.NX) JP1=1
      B(1,J)=1.
      B(2,J)=0.
      B(3,J)=0.
      B(4,J)=1.
      C(1,J)=FX*US(J,K)
      C(2,J)=FX*PHIS(J,K)
      C(3,J)=C(2,J)
      C(4,J)=C(1,J)
      DO 118 L=1,4
  118 A(L,J)=-C(L,J)
      E(1,J)=U(J,K)-FU* VS(J,K)*(U(J,KP1)-U(J,KM1))+FK*V(J,K)
      E(2,J)=PHI(J,K)-FU* PHIS(J,K)*(V(J,KP1)-V(J,KM1))-VS(J,K)*FU*
     1 (PHI(J,KP1)-PHI(J,KM1))
      IF(IND.EQ.0) GO TO 121
      E(1,J)=E(1,J)+EVX(J,K)
      GO TO 120
  121 E(1,J)=E(1,J)+BDJ*(U(JP1,K)-2.*  U(J,K)+  U(JM1,K))
      E(2,J)=E(2,J)+BDJ*(PHI(JP1,K)-2.*PHI(J,K)+PHI(JM1,K))
      IF(K.EQ.1.OR.K.EQ.NY) GO TO 120
      E(1,J)=E(1,J)+ADJ*(U(J,KP1)-2.*U(J,K)+U(J,KM1))
      E(2,J)=E(2,J)+ADJ*(PHI(J,KP1)-2.*PHI(J,K)+PHI(J,KM1))
  120 CONTINUE
      CALL CYCBLK(NX,A,B,C,E,W)
      DO 125 J=1,NX
      UBS(J,K)=E(1,J)
  125 PHIBS(J,K)=E(2,J)
C      LOOP FOR EACH POINT  X(J), J=1,...,NX  ON LINE  Y(K).
      IF(K.EQ.1) GO TO 136
      IF(K.EQ.NY) GO TO 136
      DO 130 J=1,NX
C        SET UP ELEMENTS OF CYCLIC TRIDIAGONAL MATRIX, AND CONSTANT
C              TERMS, FOR DETERMINATION OF  VBS.
      Q(J)=1.00
      R(J)=FX*US(J,K)
      P(J)=-R(J)
      S(J)=V(J,K)-FU*VS(J,K)*(V(J,KP1)-V(J,KM1))-FU*PHIS(J,K)*
     1 (PHI(J,KP1)-PHI(J,KM1))-FK*UBS(J,K)
      IF(IND.EQ.0) GO TO  131
      S(J)=S(J)+EVY(J,K)
      GO TO 130
  131 S(J)=S(J)+ADJ*(V(J,KP1)-2.*V(J,K)+V(J,KM1))
      S(J)=S(J)+BDJ*(V(JP1,K)-2.*V(J,K)+V(JM1,K))
  130 CONTINUE
C        CALL CYCTRID TO SOLVE FOR VALUES OF VBS AT ALL POINTS X(J),
C             J=1,...,NX  ON THE LINE  Y(K).
      CALL CYCTRID(NX,P,Q,R,S,W)
      DO 135 J=1,NX
  135 VBS(J,K)=S(J)
      GO TO 140
  136 DO 138 J=1,NX
  138 VBS(J,K)=0.
  140 CONTINUE
      IF(IND.EQ.0) GO TO 142
      CALL EDDY(UBS,VBS,EVX,EVY,LX,NX,NY)
C     LOOP FOR EACH VERTICAL LINE  X(J), J=1,...,NX
  142 DO 190 J=1,NX
      JP1=J+1
      JM1=J-1
      IF(JM1.LT.1) JM1=NX
      IF(JP1.GT.NX) JP1=1
C       LOOP FOR EACH POINT  Y(K), K=1,...,NY  ON LINE  X(J).
      DO 170 K=1,NY
      FU=FY
      KP1=K+1
      KM1=K-1
      IF(KP1.LE.NY) GO TO 145
      KP1=NY
      GO TO 150
  145 IF(KM1.GE.1) GO TO 155
      KM1=1
  150 FU=FU+FU
C        SET UP ELEMENTS OF BLOCK (2*2) TRIDIAGONAL MATRIX, AND CONSTANT
C              TERMS, FOR DETERMINATION OF  VN AND PHIN.
  155 B(1,K)=1.
            B(4,K)=1.
      B(2,K)=0.
      B(3,K)=0.
      C(1,K)=FU*VS(J,K)
      C(2,K)=FU*PHIS(J,K)
      C(3,K)=C(2,K)
       C(4,K)=C(1,K)
      DO 160 L=1,4
  160 A(L,K)=-C(L,K)
       E(1,K)=2.*VBS(J,K)-V(J,K)+FU      *VS(J,K)*(V(J,KP1)-V(J,KM1))+FU*
     1  PHIS(J,K)*(PHI(J,KP1)-PHI(J,KM1))
      E(2,K)=2.*PHIBS(J,K)-PHI(J,K)+FU* PHIS(J,K)*(V(J,KP1)-V(J,KM1))+FU
     1  *VS(J,K)*(PHI(J,KP1)-PHI(J,KM1))
      IF(IND.EQ.0) GO TO 170
      E(1,K)=E(1,K)+EVY(J,K)
  170 CONTINUE
      DO 172 L=1,4
      B(L,1)=B(L,1)+A(L,1)
  172 B(L,NY)=B(L,NY)+C(L,NY)
C       CALL BLKTRID TO SOLVE FOR VALUES OF VN, PHIN AT ALL POINTS Y(K),
C            K=1,...,NY  ON THE LINE   X(J).
      CALL BLKTRID(NY,1,A,B,C,E,NY)
      DO 175 K=1,NY
      VN(J,K)=E(1,K)
  175 PHIN(J,K)=F(2,K)
      VN(J,1)=0.
      VN(J,NY)=0.
 1755 CONTINUE
C        LOOP FOR EACH POINT  Y(K), K=1,...,NY  ON LINE  X(J).
      DO 180 K=1,NY
C           SET UP ELEMENTS OF TRIDIAGONAL MATRIX, AND CONSTANT TERMS, FOR
C                  DETERMINATION OF UN.
      FU=FY
      KP1=K+1
      KM1=K-1
      IF(KP1.LE.NY) GO TO 176
      KP1=NY
      GO TO 177
  176 IF(KM1.GE.1) GO TO 178
      KM1=1
  177 FU=FU+FU
  178 FK=F(K)
      Q(K)=1.
      R(K)=FU*VS(J,K)
      P(K)=-R(K)
      S(K)=2.*UBS(J,K)-U(J,K)+FU* VS(J,K)*(U(J,KP1)-U(J,KM1))
     1      -FK*(V(J,K)-VN(J,K))
      IF(IND.EQ.0) GO TO 180
      S(K)=S(K)+EVX(J,K)
  180 CONTINUE
      Q(1)=Q(1)+P(1)
      Q(NY)=Q(NY)+R(NY)
      P(1)=0.
      R(NY)=0.
C        CALL TRIDIAG TO SOLVE FOR VALUES OF UN AT ALL POINTS Y(K),
C             K=1,...,NY  ON THE LINE  X(J).
      CALL TRIDIAG(NY,1,P,Q,R,S,NY)
      DO 185 K=1,NY
  185 UN(J,K)=S(K)
  190 CONTINUE
C     ESTABLISH NEW VALUES OF  US, VS, PHIS, U, V, PHI.
      DO 192 K=1,NY
      DO 192 J=1,NX
      US(J,K)=0.5*(3.*UN(J,K)-U(J,K))
      VS(J,K)=0.5*(3.*VN(J,K)-V(J,K))
      PHIS(J,K)=0.25*(3.*PHIN(J,K)-PHI(J,K))
      U(J,K)=UN(J,K)
      V(J,K)=VN(J,K)
  192 PHI(J,K)=PHIN(J,K)
      RETURN
      END
       SUBROUTINE MAPPA(FUN,C,NX,NZ,LX)
       DIMENSION FUN(LX,NZ),ANS(4,116),IANS(116),NUM(10)
       DATA NUM/1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9,1H0/
  111    FORMAT(5X,2I5)
     1 FORMAT(//5X,23I5//)
     2 FORMAT(1H ,I3)
     3 FORMAT(1H ,7X,116A1)
     4 FORMAT(1H+,7X,116A1)
       K=3
       N=5
       FK=K
       FN=N
       I=0
       NY=NZ-1
       PRINT 111,NX,NY
       LEND=K
       PRINT 1, (J,J=1,NZ)
       JB=1
    10 I=I+1
       PRINT 2,I
       IP1=I+1
       IF(IP1.GT.NX) IP1=1
      DO 15 J=1,NZ
      XDIF=(FUN(IP1,J)-FUN(I,J))/FK
      JX=1+N*(J-JB)
      ANS(1,JX)=FUN(I,J)
      DO 15 L=2,LEND
   15 ANS(L,JX)=ANS(L-1,JX)+XDIF
   18 DO 20 J=1,NY
      JX=1+N*(J-JB)
      DO 20 L=1,LEND
      YDIF=(ANS(L,JX+N)-ANS(L,JX))/FN
      M1=JX+1
      M3=JX+N-1
      DO 20 M=M1,M3
   20 ANS(L,M)=ANS(L,M-1)+YDIF
      MEND=M3
      DO 50 L=1,LEND
      DO 40 M=1,MEND
      IF(ANS(L,M).GE.0.) GO TO 30
      AANS=-ANS(L,M)
      KANS=C*AANS
      KKANS=2*(KANS/2)
      IF(KANS.EQ.KKANS) GO TO 35
   25 KANS=KANS/2
      KANS=MOD(KANS,10)
      IF(KANS.EQ.0) KANS=10
      IANS(M)=NUM(KANS)
      GO TO 40
   30 KANS=C*ANS(L,M)
      KKANS=2*(KANS/2)
      IF(KANS.EQ.KKANS) GO TO 25
   35 IANS(M)=1H
   40 CONTINUE
      IF(L.GT.1) GO TO 45
      PRINT 4,(IANS(M),M=1,MEND)
      GO TO 50
   45 PRINT 3,(IANS(M),M=1,MEND)
   50 CONTINUE
      IF(I-NX) 10,55,65
   55 LEND=1
      I=I+1
      PRINT 2,I
      DO 60 J=1,NZ
      JX=1+N*(J-JB)
60        ANS(1,JX)=FUN(1,J)
      GO TO 18
   65 PRINT 1,(J,J=1,NZ)
      RETURN
      END
      SUBROUTINE UVOUT(W,NX,NY,LX)
      DIMENSION W(LX,NY)
      DIMENSION JA(32)
      DATA NOUT/3/
      DATA IND/0/
    1 FORMAT(3H0   ,22I6/)
    2 FORMAT(1X,I2,22F6.2)
      IF(IND.GT.0) GO TO 4
      IND=1
      K=0
      NX2=NX+2
      DO 3 J=1,NX2
      JA(J)=K
3      K=K+1
4     JE=0
5        JB=JE+1
      JE=MIN0(NX2,JE+22)
      WRITE(NOUT,1) (JA(J),J=JB,JE)
      KK=NY.
     .IF(JB.GT.1) GO TO 9
      JE=JE-1
      DO 7 K=1,NY
      KM=KK-1
      WRITE(NOUT,2)KM, W(NX,KK),(W(J,KK),J=JB,JE)
7     KK=KM
      JE=JE+1
      GO TO 5
9     JE=JE-2
      JB=JB-1
      DO 10 K=1,NY
      KM=KK-1
      WRITE(NOUT,2) KM,(W(J,KK),J=JB,JE),W(1,KK)
10    KK=KM
      RETURN
       END
      SUBROUTINE HOUT(H,NX,NY,LX)
      DIMENSION H(LX,NY)
      DIMENSION JA(32)
      DATA NOUT/3/
      DATA IND/0/
    6 FORMAT("0 HEIGHT VALUES"/)
    7 FORMAT(3X,22I6/)
    8 FORMAT(1X,I2,22F6.0)
      IF(IND.GT.0) GO TO 4
       IND=1
      K=0
      NX2=NX+2
      DO 3 J=1,NX2
      JA(J)=K
3      K=K+1
4     JE=0
5     JB=JE+1
      JE=MIN0(NX2,JE+22)
      WRITE(NOUT,6)
      WRITE(NOUT,7) (JA(J),J=JB,JE)
      KK=NY
      IF(JB.GT.1) GO TO 9
      JE=JE-1
      DO 17 K=1,NY
       KM=KK-1
      WRITE(NOUT,8) KM,H(NX,KK),(H(J,KK),J=JB,JE)
17      KK=KM
      JE=JE+1
      GO TO 5
9       JE=JE-2
       JB=JB-1
      DO 10 K=1,NY
      KM=KK-1
      WRITE(NOUT,6)KM,(H(J,KK),J=JB,JE),H(1,KK)
10        KK=KM
       RETURN
       END
       SUBROUTINE SETUP(U,V,PHI,H,F,NX,NY,S,C,LX)
C   H(X,Y)=H0+H1*TANH(P)+H2*SIN(Q)*(SECH(R))**2,
C      WHERE   P = 9.*(D/2-Y)/(2.*D),
C       AND    Q = TUPI*X/FL ,   AND  R = 2*P.
C   PHI(J,K)=2.*SQRT(G*H(J,K))
C     U(J,K)=-(G/F(K))*(PARTIAL DERIVATIVE DH/DY AT J,K)
C     V(J,K)= (G/F(K))*(PARTIAL DERIVATIVE DH/DX AT J,K)
       COMMON/CONST/FL,D,T,DX,DY,DT,FX,FY,FT,G,TIME,IPR,ADJ,BDJ
       DIMENSION  U(LX,NY),V(LX,NY),PHI(LX,NY),F(NY),S(LX),C(LX),H(LX,NY)
       DATA TUPI/6.2831853071796/
       DATA NIN/1/
       DATA NOUT/3/
     1 FORMAT(6E10.4)
     3 FORMAT("1 SHALLOW WATER EQUATIONS"/)
     4 FORMAT("0 CONSTANTS:  H0=",F5.0," M",10X,"FHAT=",E9.2,"/SEC  ",
      1 10X,"L=",F9.0," M",12X,"DX=",F8.0," M"/14X,"H1=",F5.0," M",10X,
      2 "BETA=",E9.2,"/SEC/M",10X,"D=",F9.0," M",12X,"DY=",F8.0," M"/
      3 14X,"H2=",F5.0," M",40X,"T=",F9.0," SEC",10X,"DT=",F8.0," SEC"/)
```

```
C    H0, H1, H2  ARE CONSTANTS IN THE HEIGHT FUNCTION
C    FHAT, BETA  ARE CONSTANTS IN    F = FHAT + BETA*(Y-D/2)
      READ(NIN,1) H0,H1,H2,FHAT,BETA
      WRITE(NOUT,3)
      WRITE(NOUT,4) H0,FHAT,FL,DX,H1,BETA,D,DY,H2,T,DT
      YE=9./D
      YF=0.5*YE
      D2=D/2.
      XF=TUPI/FL
      FNXI=TUPI/FLOAT(NX)
    8 FJ=0.
      DO 10 J=1,NX
      FJ=FJ+1.
      TEMP=FJ*FNXI
      S(J)=SIN(TEMP)
   10 C(J)=COS(TEMP)
      S(NX)=0.
      C(NX)=1.
      NYM=NY-1
      FNYMI=9./FLOAT(NYM)
      FKM=0.
      Y=0.
      DO 20 K=1,NY
      TEMP=D2-Y
      F(K)=FHAT-BETA*TEMP
      GH= G/F(K)
      YA=4.5-FKM*FNYMI
      YB=0.5*YA
      TNH=TANH(YB)
      SH2=1.-TNH*TNH
      C1=H0+H1*TNH
      C4=-YF*SH2*H1
      TNH=TANH(YA)
      SH2=1.-TNH*TNH
      C2=H2*SH2
      IF(K.EQ.1) C2=0.
      IF(K.EQ.NY) C2=0.
      C3=C2*XF
      C5=2.*C2*YE*TNH
      DO 15 J=1,NX
      TEMP=S(J)
      H(J,K)=C1+C2*TEMP
      PHI(J,K)=2.*SQRT(G*H(J,K))
   14 V(J,K)=GH*C3*C(J)
      U(J,K)=-GH*(C4+C5*TEMP)
   15 CONTINUE
      F(K)=F(K)*FT
      Y=Y+DY
   20 FKM=FKM+1.
   24 DO 25 J=1,NX
      V(J,1)=0.
   25 V(J,NY)=0.
      RETURN
      END
      SUBROUTINE LOOK(U,V,PHI,H,F,NX,NY,LX)
      REAL MSVRT
      COMMON/CONST/FL,D,T,DX,DY,DT,FX,FY,FT,G,TIME,IPR,ADJ,BDJ
      DIMENSION U(LX,NY),V(LX,NY),PHI(LX,NY),H(LX,NY),F(NY)
      DATA NOUT/3/
      DATA IND/0/,NSTEP/0/,TIMEA/0./
    2 FORMAT("1")
    3 FORMAT("1 TIME=",F9.0," SEC",10X,"HMEAN=",F8.2," M",10X,"ENERGY=",
     1 1PE13.6,10X,"CPU TIME FOR ",I3," STEPS =",0PF8.2," SEC")
 4     FORMAT(2X,*MEAN SQUARE VORTICITY=*,1PE13.6)
 55   FORMAT(5X,"LOOK")
      TIMEB=SECOND(CPU)
      DPTIME=TIMEB-TIMEA
      IF(IND.GT.0) GO TO 5
      G4INV=1./(4.*G)
      AREA=NX*(NY-1)
      ECNST=DX*DY/(G+G)
    5 SUMENG=0.
      HMEAN=0.
      ZMEAN=0.
       ECNST2=DX*DY
      NY1=NY-1
      FAC=0.5
      DO 40 K=1,NY
      IF(K.EQ.NY) FAC=0.5
      MSVRT=0.
      HEL=0.
      ENEREL=0.
      DO 10 J=1,NX
      PHSQ=PHI(J,K)*PHI(J,K)/4.
      ENEREL=PHSQ*(PHSQ+U(J,K)*U(J,K)+V(J,K)*V(J,K))+ENEREL
   10 CONTINUE
      IF(IND.GT.0) GO TO 20
      DO 15 J=1,NX
   15 HEL=HEL+H(J,K)
      GO TO 30
   20 DO 25 J=1,NX
      H(J,K)=PHI(J,K)*PHI(J,K)*G4INV
   25 HEL=HEL+H(J,K)
   30 IF(FAC.EQ.1) GO TO 35
      HEL=HEL*FAC
   35 HMEAN=HMEAN+HEL
      SUMENG=SUMENG+ENEREL
   40 FAC=1.0
      DO 60 K=2,NY1
      F(K)=F(K)/FT
       DO 56 J=1,NX
      JP1=J+1
      JM1=J-1
      IF(JM1.LT.1   ) JM1=NX
      IF(JP1.GT.NX) JP1=1
      MSVRT=(((V(JP1,K)-V(JM1,K))/(2.*DX)-(U(J,K+1)-U(J,K-1))/(2.*DY)+
     1F(K))**2)/H(J,K)+MSVRT
 56    CONTINUE
      ZMEAN=ZMEAN+MSVRT
 60   CONTINUE
      HMEAN=HMEAN/AREA
      ENERGY=SUMENG*ECNST
       ZMEAN=ZMEAN*ECNST2
      WRITE(NOUT,3) TIME,HMEAN,ENERGY,NSTEP,DPTIME
       WRITE(3,4) ZMEAN
      NSTEP=IPR
      CALL HOUT(H,NX,NY,LX)
      WRITE(NOUT,2)
      CALL MAPPA(H,0.02,NX,NY,LX)
      TIMEA=SECOND(CPU)
      IF(IND.NE.0) GO TO 45
      EN2=ENERGY+ENERGY
      IND=1
      GO TO 50
   45 IF(ENERGY.GT.EN2) STOP
      WRITE(NOUT,55)
   50 RETURN
      END
      SUBROUTINE CYCBLK(N,P,Q,R,D,W)

      DIMENSION P(4,N),Q(4,N),R(4,N),D(2,N),W(2,N,3)

C  CYCBLK SOLVES  T*X=D  WHERE T IS A DIAGONALLY DOMINANT CYCLIC BLOCK
C     TRIDIAGONAL MATRIX, EACH BLOCK BEING  2*2, AND THE ORDER OF T BEING 2*N.
C  P, Q, R, ARE ARRAYS OF DIMENSION 4*N, WHICH CONTAIN THE ELEMENTS OF
C    THE BLOCKS ALONG THE SUB-DIAGONAL, DIAGONAL AND SUPER-DIAGONAL
C     RESPECTIVELY.
C  D  IS AN ARRAY DIMENSIONED  2*N, CONTAINING THE CONSTANTS.
```

```
C  THE BLOCK IN POSITION  T(1,N)  IS STORED IN  P( ,1), AND
C       THAT IN POSITION  T(N,1)  IS STORED IN  R( ,N).

C  ARRAY W PROVIDES WORKING AREA AND MUST BE OF LENGTH 2*N*3 AT LEAST.
C  DURING COMPUTATION ALL INPUT ARRAYS ARE OVER-WRITTEN.

C  THE SOLUTION VECTORS ARE STORED IN ARRAY  D.
      NM=N-1

      DO 10 I=1,NM

      DO 5 J=1,2
      W(J,I,1)=D(J,I)
      W(J,I,2)=0.
    5 W(J,I,3)=0.
   10 CONTINUE

      W(1,1,2)=P(1,1)
      W(2,1,2)=P(2,1)
      W(1,1,3)=P(3,1)
      W(2,1,3)=P(4,1)
      W(1,NM,2)=R(1,NM)
      W(2,NM,2)=R(2,NM)
      W(1,NM,3)=R(3,NM)
      W(2,NM,3)=R(4,NM)

   CALL BLKTRID(NM,3,P,Q,R,W,N)

   V1= Q(4,N)-R(2,N)*W(1,1,3)-R(4,N)*W(2,1,3)-P(2,N)*W(1,NM,3)
  1    -P(4,N)*W(2,NM,3)
   V2=-Q(2,N)+R(2,N)*W(1,1,2)+R(4,N)*W(2,1,2)+P(2,N)*W(1,NM,2)
  1    +P(4,N)*W(2,NM,2)
   V3=-Q(3,N)+R(1,N)*W(1,1,3)+R(3,N)*W(2,1,3)+P(1,N)*W(1,NM,3)
  1    +P(3,N)*W(2,NM,3)
   V4=Q(1,N)-R(1,N)*W(1,1,2)-R(3,N)*W(2,1,2)-P(1,N)*W(1,NM,2)-P(3,N)*
  1     W(2,NM,2)
   D1N     =D(1,N)-R(1,N)*W(1,1,1)-R(3,N)*W(2,1,1)-P(1,N)*W(1,NM,1)
  1          -P(3,N)*W(2,NM,1)
   D2N     =D(2,N)-R(2,N)*W(1,1,1)-R(4,N)*W(2,1,1)-P(2,N)*W(1,NM,1)
  1          -P(4,N)*W(2,NM,1)
   DET=V1*V4-V2*V3
   TEM=(V1*D1N+V3*D2N)/DET
   D2N=(V2*D1N+V4*D2N)/DET
   D1N=TEM

   DO 20 I=1,NM
   D(1,I)=W(1,I,1)-D1N*W(1,I,2)-D2N*W(1,I,3)
   D(2,I)=W(2,I,1)-D1N*W(2,I,2)-D2N*W(2,I,3)
20 CONTINUE

   D(1,N)=D1N
   D(2,N)=D2N
   RETURN

   END

      SUBROUTINE CYCTRID(N,P,Q,R,D,W)


      DIMENSION P(N),Q(N),R(N),D(N),W(N,2)

C  SOLVES THE SET  A*X=D  OF N (N.GE.3) LINEAR EQUATIONS WHERE THE
C    COEFFICIENT MATRIX IS CYCLIC TRIDIAGONAL.
C  VECTORS  P, Q, R, D, EACH OF N ELEMENTS,CONTAIN RESPECTIVELY THE
C    SUB-DIAGONAL, DIAGONAL, SUPER-DIAGONAL, AND CONSTANT ELEMENTS.
C    THE ELEMENT  A(1,N)  IS STORED IN P(1),
C             AND  A(N,1)  IS STORED IN R(N).

C  ARRAY W PROVIDES WORKING AREA AND MUST BE OF LENGTH 2*N AT LEAST.
C  DURING COMPUTATION ALL INPUT ARRAYS ARE OVER-WRITTEN.
```

```
C   THE SOLUTION VECTOR IS STORED IN VECTOR  D.
      NM=N-1

      DO 10 I=1,NM
      W(I,1)=Q(I)
   10 W(I,2)=0.
      W(1,2)=P(1)
      W(NM,2)=R(NM)

      CALL TRIDIAG(NM,2,P,Q,R,W,N)

      XN=(D(N)-R(N)*W(1,1)-P(N)*W(NM,1))/(Q(N)-R(N)*W(1,2)-P(N)*W(NM,2))

      DO 20 I=1,NM
   20 D(I)=W(I,1)-XN*W(I,2)
      D(N)=XN
      RETURN
      END

      SUBROUTINE BLKTRID(N,M,A,B,C,D,L)

C   BLKTRID SOLVES  T*X=D  WHERE T IS A DIAGONALLY DOMINANT TRIDIAGONAL
C     BLOCK MATRIX, EACH BLOCK BEING  2*2, AND THE ORDER OF T BEING 2*N.
C   DURING COMPUTATION ALL INPUT ARRAYS ARE OVER-WRITTEN.

C   THE SOLUTION VECTORS REPLACE D, THE ARRAY OF CONSTANTS.

      DIMENSION A(4,L),B(4,L),C(4,L),D(2,L,M)
      IF(M.EQ.3) GO TO 5
      A(1,2)=0.
      A(2,2)=0.
      B(2,N)=0.
      B(2,1)=0.
      C(1,N-1)=0.
      C(2,N-1)=0.
    5 TEMP=1./(B(1,1)*B(4,1)-B(2,1)*B(3,1))
      V1=B(4,1)*TEMP
      V2=-B(2,1)*TEMP
      V3=-B(3,1)*TEMP
      V4=B(1,1) *TEMP

      DO 10 J=1,M
      TEMP= V1*D(1,1,J)+V3*D(2,1,J)
      D(2,1,J)= V2*D(1,1,J)+V4*D(2,1,J)
   10 D(1,1,J)=TEMP
      TEMP=-(V1*C(1,1)+V3*C(2,1))
      C(2,1)=-(V2*C(1,1)+V4*C(2,1))
      C(1,1)=TEMP
      TEMP=-(V1*C(3,1)+V3*C(4,1))
      C(4,1)=-(V2*C(3,1)+V4*C(4,1))
      C(3,1)=TEMP

      DO 20 I=2,N
      IM=I-1
      B(1,I)=B(1,I)+A(1,I)*C(1,IM)+A(3,I)*C(2,IM)
      B(2,I)=B(2,I)+A(2,I)*C(1,IM)+A(4,I)*C(2,IM)
      B(3,I)=B(3,I)+A(1,I)*C(3,IM)+A(3,I)*C(4,IM)
      B(4,I)=B(4,I)+A(2,I)*C(3,IM)+A(4,I)*C(4,IM)
      TEMP=1./(B(1,I)*B(4,I)-B(2,I)*B(3,I))
      V1=B(4,I) *TEMP
      V2=-B(2,I)*TEMP
      V3=-B(3,I)*TEMP
      V4=B(1,I) *TEMP
      TEMP=-(V1*C(1,I)+V3*C(2,I))
      C(2,I)=-(V2*C(1,I)+V4*C(2,I))
      C(1,I)=TEMP
      TEMP=-(V1*C(3,I)+V3*C(4,I))
      C(4,I)=-(V2*C(3,I)+V4*C(4,I))
      C(3,I)=TEMP
      DO 15 J=1,M
      D(1,I,J)=D(1,I,J)-A(1,I)*D(1,IM,J)-A(3,I)*D(2,IM,J)
      D(2,I,J)=D(2,I,J)-A(2,I)*D(1,IM,J)-A(4,I)*D(2,IM,J)
      TEMP= V1*D(1,I,J)+V3*D(2,I,J)
      D(2,I,J)= V2*D(1,I,J)+V4*D(2,I,J)
      IF(M.EQ.1.AND.I.EQ.N) TEMP=0.
   15 D(1,I,J)=TEMP

   20 CONTINUE
      K=N
      DO 30 I=2,N
      KP=K
      K=K-1

      DO 25 J=1,M
      TEMP    =D(1,K,J)+C(1,K)*D(1,KP,J)+C(3,K)*D(2,KP,J)
      D(2,K,J)=D(2,K,J)+C(2,K)*D(1,KP,J)+C(4,K)*D(2,KP,J)
      IF(M.EQ.1.AND.K.EQ.1) TEMP=0.
   25 D(1,K,J)=TEMP

   30 CONTINUE

      RETURN

      END
      SUBROUTINE TRIDIAG(N,M,A,B,C,D,L)


      DIMENSION  A(N),B(N),C(N),D(L,M)

      DO 10 J=1,M
   10 D(1,J)=D(1,J)/B(1)
      C(1)=-C(1)/B(1)

      DO 20 I=2,N
      IM=I-1
      B(I)=A(I)*C(IM)+B(I)
      C(I)=-C(I)/B(I)
      DO 15 J=1,M
   15 D(I,J)=(D(I,J)-A(I)*D(IM,J))/B(I)
   20 CONTINUE

      K=N
      DO 30 I=2,N
      KP=K
      K=K-1
      DO 25 J=1,M
   25 D(K,J)=D(K,J)+C(K)*D(KP,J)
   30 CONTINUE

      RETURN
      END
      SUBROUTINE EDDY(U,V,EVX,EVY,LX,NX,NY)
      COMMON/EDVOPT/GNU,GAM,GDX3,PX,PY,PXSQ,PYSQ,PXS2,PYS2,ALPHA,FAC,IND
      DIMENSION U(LX,NY),V(LX,NY),EVX(LX,NY),EVY(LX,NY),XSI(35,25),
     1 VISC(35,25)
   20 DO 25 K=1,NY
      KP1=K+1
      IF(KP1.GT.NY) KP1=NY
      DO 25 J=1,NX
      JP1=J+1
      IF(JP1.GT.NX) JP1=1
      XSI(J,K)=PX*(V(JP1,KP1)-V(J,K)+V(JP1,K)-V(J,KP1))
     1          -PY*(U(JP1,KP1)-U(J,K)+U(J,KP1)-U(JP1,K))
   25 CONTINUE
      DO 30 K=1,NY
      KP1=K+1
      IF(KP1.GT.NY) KP1=NY
      DO 30 J=1,NX
      JP1=J+1
      IF(JP1.GT.NX) JP1=1
      TA=XSI(JP1,KP1)-XSI(J,K)+XSI(JP1,K)-XSI(J,KP1)
      TB=XSI(JP1,KP1)-XSI(J,K)+XSI(J,KP1)-XSI(JP1,K)
      SQ=SQRT(PXSQ*TA*TA+PYSQ*TB*TB)
      IF(IND.EQ.2) GO TO 28
      VISC(J,K)=GNU*(1.+GDX3*SQ)
      GO TO 30
   28 VISC(J,K)=FAC*SQ
   30 CONTINUE
      DO 55 K=1,NY
      KP1=K+1
      KP2=K+2
      IF(KP1-NY) 40,37,35
   35 KP1=NY
   37 KP2=NY
   40 DO 55 J=1,NX
      JP1=J+1
      JP2=J+2
      IF(JP1-NX) 50,47,45

   45 JP1=1
      JP2=2
      GO TO 50
   47 JP2=1
   50 EVX(J,K)=PXS2*((VISC(JP2,K)+VISC(JP1,K))*(U(JP2,K)-U(JP1,K))-
     1 (VISC(JP1,K)+VISC(J,K))*(U(JP1,K)-U(J,K)))
     2         -PYS2*((VISC(J,KP2)+VISC(J,KP1))*(U(J,KP2)-U(J,KP1))-
     3 (VISC(J,KP1)+VISC(J,K))*(U(J,KP1)-U(J,K)))
      EVY(J,K)=PXS2*((VISC(JP2,K)+VISC(JP1,K))*(V(JP2,K)-V(JP1,K))-
     1 (VISC(JP1,K)+VISC(J,K))*(V(JP1,K)-V(J,K)))
     2         -PYS2*((VISC(J,KP2)+VISC(J,KP1))*(V(J,KP2)-V(J,KP1))-
     3 (VISC(J,KP1)+VISC(J,K))*(V(J,KP1)-V(J,K)))
   55 CONTINUE
      RETURN
      END
```