Computers & Geosciences Vol. 12, No. 2, pp. 151–173, 1986
Printed in Great Britain.

0098-3004/86 $3.00 + .00
© 1986 Pergamon Press Ltd.

GUSTAF: A QUASI-NEWTON NONLINEAR ADI FORTRAN IV PROGRAM FOR SOLVING THE SHALLOW-WATER EQUATIONS WITH AUGMENTED LAGRANGIANS

I. M. NAVON[†] and R. DE VILLIERS
National Research Institute for Mathematical Sciences, CSIR, PO Box 395, Pretoria 0001, South Africa

(*Received* 11 *February* 1985; *revised* 3 *September* 1985)

Abstract—A FORTRAN IV computer program is documented which implements the nonlinear alternating direction implicit (ADI) method of Gustafsson (1971) for a limited area finite-difference integration of a shallow-water equations model on a β-plane. In this method a computationally efficient quasi-Newton method is used to solve, at each time-step, the resulting nonlinear systems of algebraic equations. Large time-steps can be employed with this method, which is stable unconditionally for the linearized equations. Owing to its nonlinearity, the method is useful particularly where accuracy is important. An augmented Lagrangian method is applied to enforce conservation of the integral invariants of the shallow-water equations. This method approximates the nonlinearly constrained minimization problem by solving a series of unconstrained minimization problems.

Program options include a line-printer plot of the height-field contour and determination, at each time-step, of the three integral invariants of the shallow-water equations. According to the number of nonlinear quasi-Newton (QN) iterations performed at each time-step, different QN methods are presented. Long-term runs have been performed using this program and, due to the enforcement of conservation of integral-invariants via the augmented Lagrangian method, no finite-time "blowing" was experienced.

Key Words: Shallow-water equations, Alternating Direction Implicit (ADI), Quasi-Newton solution of nonlinear equations, Augmented Lagrangian nonlinear optimization, Conservation of integral invariants.

INTRODUCTION

The shallow-water equations, that is the barotropic primitive equations for an incompressible, inviscid fluid with a free surface, constitute a quasi-linear system of hyperbolic partial differential equations.

When discretized by explicit time difference approximations, these equations are subjected to the Courant–Friedrichs–Levy (CFL) stability condition, which severely restricts the time-step. In oceanographic and meteorological applications the discretization error in time is small compared with the discretization error in space, and the short time-step constraint imposed by the CFL condition is thus particularly annoying; it can be avoided by using implicit time differencing schemes.

Gustafsson (1971) proposed an efficient, fully implicit nonlinear alternating-direction scheme for solving the shallow-water equations, his method being based on a scheme first proposed by Kreiss and Widlund (1966).

The method necessitates the solution of a number of nonlinear systems of algebraic equations at each time-step of the numerical integration. Owing to the use of a quasi-Newton method, however, considerable computational efficiency is achieved.

In the first section of this paper, a review is given of the Gustafsson ADI algorithm applied to the shallow-water equations on a β-plane; this review is followed by a description of the quasi-Newton method for solving nonlinear systems of algebraic equations, along with the application of the method to Gustafsson's ADI algorithm.

[†]Present Address: Supercomputer Computations Research Institute, The Florida State University, Tallahasse, Florida 32306, U.S.A.

In the next section the augmented Lagrangian method and its algorithmic implementation are detailed.

The remainder of this paper is devoted to a description of the program GUSTAF, illustrated by a test problem, and to specifications for its use. A listing of the FORTRAN IV source code of the program GUSTAF is included in the Appendix. Typical outputs for 48-hour forecasts also are presented in the Appendix, along with long-term runs (including a dissipation factor) for 20 days of forecasts. Printer-plotted maps of the height field are included for different QN methods, differing according to the number of nonlinear iterations performed at each time-step.

REVIEW OF THE GUSTAFSSON NONLINEAR ADI METHOD

The shallow-water equations

The shallow-water equations can be written in vector form (Houghton, Kasahara, and Washington, 1966) as follows:

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

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

$$w = (u, v, \Phi)^T; \quad (2)$$

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

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

where h is the depth of the fluid and g the acceleration of gravity. A, B, and C are the matrices

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

$$\text{and } C = \begin{bmatrix} 0 & f & 0 \\ -f & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{4}$$

where f is the "Coriolis term" given by

$$f = \hat{f} + \beta(y - D/2) \qquad \beta = \frac{\partial f}{\partial y}, \tag{5}$$

with $\hat{f}$ 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 these boundary conditions and with the initial condition

$$w(x,y,0) = \varphi(x,y), \tag{8}$$

the total energy

$$E = \frac{1}{2}\int_0^L\int_0^D (u^2 + v^2 + \phi^2/4)\phi^2/4_g \, dx\, dy \tag{9}$$

is independent of time.

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

$$\bar{h} = 1/\bar{A}\int_0^L\int_0^D h \, dx\, dy \tag{10}$$

is independent of time.

$\bar{A}$ is the surface of the integration domain.

Applications

Although the shallow-water equations are simpler than the 3-D primitive equations describing the atmosphere — some essential numerical aspects of large-scale prediction equations are preserved. The problem of numerically solving this set of equations is similar to that of solving the hydrostatic primitive equations, because the same mixture of fast and slow motions occurs. Consequently, investigators usually use the simpler set of equations to test new numerical weather-prediction schemes. The purpose of the method exposed here is to present an accurate scheme to solve the nonlinear shallow-water equations along with a new technique to enforce a posteriori conservation of the integral invariants of the shallow-water equations in the discretized solution. This ensures the long-term accuracy of the method (see also the Appendix).

The nonlinear Gustafsson ADI algorithm

Let N_x and N_y be positive integers and set

$$\Delta x = L/N_x, \qquad \Delta y = D/N_y. \tag{11}$$

We shall denote by $w_{jk}^n (j = 0, 1, \ldots N_x;\ k = 0, 1, \ldots N_y;\ n = 0, 1, \ldots)$ an approximation to $w(j\,\Delta x, k\,\Delta y, n\,\Delta t)$, where Δt is the time-step. The basic difference operators are

$$\begin{aligned} D_{0x} w_{jk}^n &= (w_{j+1,k}^n - w_{j-1,k}^n)/(2\,\Delta x), \\ D_{+x} w_{jk}^n &= (w_{j+1,k}^n - w_{jk}^n)/\Delta x, \\ D_{-x} w_{jk}^n &= (w_{jk}^n - w_{j-1,k}^n)/\Delta x \end{aligned} \tag{12}$$

respectively, with similar definitions for D_{0y}, D_{+y}, and D_{-y}. We also define the operators P_{jk}^n and Q_{jk}^n by

$$\begin{aligned} P_{jk}^n &= \Delta t/2(A(w_{jk}^n)D_{0x} + C_k^{(1)}), \\ Q_{jk}^n &= \Delta t/2(B(w_{jk}^n)D_k + C_k^{(2)}), \end{aligned} \tag{13}$$

with

$$D_k = \begin{cases} D_{0y} & k = 1, 2, \ldots, N_y - 1; \\ D_{+y} & k = 0; \\ D_{-y} & k = N_y \end{cases} \tag{14}$$

(owing to boundary conditions in the y direction)

$$C_k^{(1)} = \begin{bmatrix} 0 & 0 & 0 \\ -f_k & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad C_k^{(2)} = \begin{bmatrix} 0 & f_k & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{15}$$

Then Gustafsson's nonlinear ADI difference scheme is defined by

$$(I - P_{jk}^{n+(1/2)})w_{jk}^{n+(1/2)} = (I + Q_{jk}^n)w_{jk}^n, \tag{16a}$$

$$(I - Q_{jk}^{n+1})w_{jk}^{n+1} = (I + P_{jk}^{n+1})w_{jk}^{n+(1/2)}. \tag{16b}$$

These equations do not apply to the v component for $k = 0$, $k = N_y$, but we use the conditions

$$v_{j0}^n = v_{j,N_y}^n = 0, \qquad n = 0, 1, \ldots \tag{17}$$

From Equations (16a) and (16b) it is clear that nonlinear systems of algebraic equations have to be solved at each time-step. For proofs of stability and accuracy see Gustafsson (1971).

The Quasi-Newton Method

The nonlinear system of algebraic equations is written in the form

$$g(\alpha) = 0, \quad (18)$$

where α is the vector of unknowns.

In our situation, owing to the fact that not more than two variables are coupled to each other on the left-hand sides of Equations (16), one first solves equation (16a) for

$$\alpha = (u_1, \Phi_1, u_2, \Phi_2, \ldots, \Phi_{N_x}), \quad (19)$$

omitting the n and k indices for simplicity of notation. The Newton method, described, for example, in Isaacson and Keller (1966), is given by

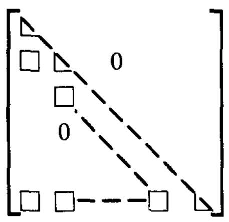
$$\alpha^{(m+1)} = \alpha^{(m)} - J^{-1}(\alpha^{(m)})g(\alpha^{(m)}), \quad (20)$$

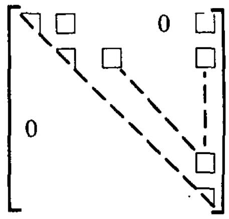
where the superscript denotes the iteration and J is the Jacobian

$$J = \partial(g, \alpha) = \left(\frac{\partial g}{\partial \alpha}\right). \quad (21)$$

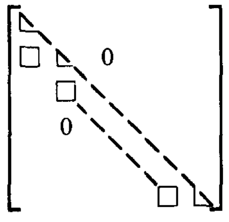
Owing to the structure of the Gustafsson algorithm for the shallow-water equations, the Jacobian matrix is either block cyclic tridiagonal or block tridiagonal.

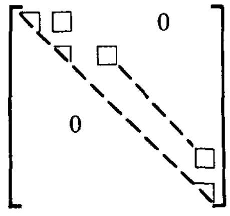
In order to solve $J^{-1}g$ in Equation (20) an LU-decomposition is applied to J (see, for example, Isaacson and Keller, 1966, chapter 2.3.3) where L and U have either the forms

$L =$ ,

$U =$  (22)

for cyclic block tridiagonal matrices or

$L =$ ,

$U =$  (23)

for block tridiagonal matrices.

The squares and the triangles in this situation indicate (2×2)-matrices. $J^{-1}g$ then is computed by backsubstitution in two stages. First, z is solved from

$$Lz = g, \quad (24)$$

and then $J^{-1}g$ is solved from

$$U(J^{-1}g) = z.$$

In the quasi-Newton method, the computationally expensive LU decomposition ($O(n^3)$ operations) is performed only once every M-th time-step, where M is a fixed integer.

Because the backsubstitution is a fast operation, the quasi-Newton method is efficient computationally, provided the number of nonlinear iterations at each time-step is small.

The quasi-Newton formula is

$$\alpha^{(m+1)} = \alpha^{(m)} - \hat{J}^{-1}(\alpha^{(m)}) \cdot g(\alpha^{(m)}), \quad (26)$$

where

$$\hat{J} = J(\alpha^{(0)}) + 0(\Delta t). \quad (27)$$

The method works when M, the number of time-steps between successive updatings of the LU decomposition of the Jacobian matrix J, is a relatively small number; in our situation, $M = 6$ or $M = 12$.

Gustafsson (1971) proves that even one quasi-Newton (QN) iteration is sufficient at each time-step; this method will be denoted by QNEX1. The QN method with two iterations per time-step is denoted by QN2, whereas the QN method with three iterations, QN3, is used for comparison and accuracy tests.

Implementation of the QN Method in the Gustafsson ADI Algorithm

(1) Solving the first intermediary ADI step in the x-direction [Eq. (16a)] and noting that not more than two variables are coupled to each other on the left-hand side of (16a), we first solve $(u_{jk}^{n+(1/2)}, \Phi_{jk}^{n+(1/2)})$, that is Equation (19) is solved:

$$\alpha = (u_1, \Phi_1, u_2, \Phi_2 \ldots \Phi_{N_x})_k^{nT}.$$

The detailed equations for $(u_{jk}^{n+1/2}, \Phi_{jk}^{n+1/2})$ are

$$\begin{aligned} u_{jk}^{n+(1/2)} &+ \frac{\Delta t}{2} u_{jk}^{n+(1/2)}(u_{j+1,k}^{n+(1/2)} - u_{j-1,k}^{n+(1/2)})/2\Delta x \\ &+ \frac{\Delta t}{2}\frac{\Phi_{jk}^{n+(1/2)}}{2}(\Phi_{j+1,k}^{n+(1/2)} - \Phi_{j-1,k}^{n+(1/2)})/(2\Delta x) \\ &= u_k^n - \frac{\Delta t}{2} v_{jk}^n(u_{j,k+1}^n - u_{j,k-1}^n)/(2\Delta y) \\ &\qquad - \frac{\Delta t}{2} f_k v_{jk}^n, \quad (28) \end{aligned}$$

$$\begin{aligned}\Phi_{jk}^{n+(1/2)} &+ \frac{\Delta t}{2}\Phi_{jk}^{n+(1/2)}(u_{j+1,k}^{n+(1/2)} - u_{j-1,k}^{n+(1/2)})/(2\Delta x)\\ &+ \frac{\Delta t}{2}u_{jk}^{n+(1/2)}(\Phi_{j+1,k}^{n+(1/2)} - \Phi_{j-1,k}^{n+(1/2)})/(2\Delta x)\\ &= \Phi_{jk}^{n} - \frac{\Delta t}{2}\Phi_{jk}^{n}(v_{j,k+1}^{n} - v_{j,k-1}^{n})/(2\Delta y)\\ &\quad - \frac{\Delta t}{2}v_{jk}^{n}(\Phi_{j,k+1}^{n} - \Phi_{j,k-1}^{n})/(2\Delta y)\,. \qquad (29)\end{aligned}$$

Using the notation

$$\lambda_x = \frac{\Delta t}{\Delta x}, \qquad \lambda_y = \frac{\Delta t}{\Delta y} \qquad (30)$$

and the definition of the Jacobian, we obtain J in the form

$$J = \begin{bmatrix} D_1 & H_1 & & & & \\ -H_2 & D_2 & H_2 & & & \\ & H_3 & D_3 & H_3 & & 0 \\ & & \ddots & \ddots & \ddots & \\ & 0 & & & \ddots & H_{N_x} - 1 \\ H_{N_x} & & & & H_{N_x} & D_{N_x} \end{bmatrix} \qquad (31)$$

where

$$H_j = \frac{\lambda_x}{8}\begin{bmatrix} 2u_j & \Phi_j \\ \Phi_j & 2u_j \end{bmatrix}_k^{n+(1/2)} \qquad (32)$$

and

$$D_j = \begin{bmatrix} 1 + \frac{\lambda_x}{4}(u_{j+1} - u_{j-1}) & \frac{\lambda_x}{8}(\Phi_{j+1} - \Phi_{j-1}) \\ \frac{\lambda_x}{4}(\Phi_{j+1} - \Phi_{j-1}) & 1 + \frac{\lambda_x}{8}(u_{j+1} - u_{j-1}) \end{bmatrix}_k^{n+(1/2)} \qquad (33)$$

The *LU* decomposition of this cyclic block tridiagonal matrix (at every M-th time-step) is performed next (see also Navon, 1977) and $J^{-1}g$ is computed by back-substitution.

(2) Once $(u^{n+(1/2)}, \Phi^{n+(1/2)})_{jk}$ are known, we determine $v_{jk}^{n+(1/2)}$ in the same way. Writing the equation for $v_{jk}^{n+(1/2)}$ in (16a), we obtain

$$\begin{aligned} v_{jk}^{n+(1/2)} &+ \frac{\Delta t}{2}u_{jk}^{n+(1/2)}(v_{j+1,k}^{n+(1/2)} - v_{j-1,k}^{n+(1/2)})/(2\Delta x)\\ &+ \frac{\Delta t}{2}f_k u_{jk}^{n+(1/2)} = v_{jk}^{n} - \frac{\Delta t}{2}v_{jk}^{n}(v_{j,k+1}^{n} - v_{j,k-1}^{n})/(2\Delta y)\\ &- \frac{\Delta t}{2}\frac{\Phi_{jk}}{2}(\Phi_{j,k-1}^{n} - \Phi_{j,k+1}^{n})/(2\Delta y)\,. \qquad (34)\end{aligned}$$

$$\alpha \text{ is now } (v_1, v_2 \ldots v_{N_x})_k^{n^T}, \qquad (35)$$

and, by performing the partial derivatives in the Jacobian matrix J, we obtain

$$D_j = 1, \qquad H_j = \frac{\lambda_x}{4}u_{jk}^{n+(1/2)}. \qquad (36)$$

The matrix J now is cyclic scalar tridiagonal. The LU decomposition is performed once again at every M-th time-step and $J^{-1}g$ is solved by backsubstitution.

(3) To solve w_{jk}^{n+1} we use the second part of the Gustafsson algorithm (16b).

We then first solve the coupled variables

$$(v_{jk}^{n+1}, \Phi_{jk}^{n+1}) \qquad (37)$$

$\alpha = (v_1, \Phi_1, v_2, \Phi_2 \ldots . \Phi_{N_y})$ fixed j and $n + 1$. As the boundary conditions are not periodic in the y direction, the J matrix now is block tridiagonal, and consequently the extra rows and columns in the L and U matrices, respectively, do not occur. To obtain the entries for the J matrix, we write the equations for v_{jk}^{n+1} and Φ_{jk}^{n+1}, respectively:

$$\begin{aligned} v_{jk}^{n+1} &+ \frac{\Delta t}{2}v_{jk}^{n+1}(v_{j,k+1}^{n+1} - v_{j,k-1}^{n+1})/(2\,\Delta y)\\ &+ \frac{\Delta t}{4}\Phi_{jk}^{n+1}(\Phi_{j,k+1}^{n+1} - \Phi_{j,k-1}^{n+1})/(2\,\Delta y) = v_{jk}^{n+(1/2)}\\ &- \frac{\Delta t}{2}u_{jk}^{n+(1/2)}(v_{j+1,k}^{n+(1/2)} - v_{j-1,k}^{n+(1/2)})/(2\,\Delta x)\\ &- f_k u_{jk}^{n+(1/2)}, \qquad (38)\end{aligned}$$

$$\begin{aligned} \Phi_{jk}^{n+1} &+ \frac{\Delta t}{2}\Phi_{jk}^{n+1}(v_{j,k+1}^{n+1} - v_{j,k-1}^{n+1})/(2\,\Delta y)\\ &+ \frac{\Delta t}{2}v_{jk}^{n+1}(\Phi_{j,k+1}^{n+1} - \Phi_{j,k-1}^{n+1})/(2\,\Delta y) = \Phi_{jk}^{n+(1/2)}\\ &- \frac{\Delta t}{2}\Phi_{jk}^{n+(1/2)}(u_{j+1,k}^{n+(1/2)} - u_{j-1,k}^{n+(1/2)})/(2\,\Delta x)\\ &- \frac{\Delta t}{2}u_{jk}^{n+(1/2)}(\Phi_{j+1,k}^{n+(1/2)} - \Phi_{j-1,k}^{n+(1/2)})/(2\,\Delta x)\,. \qquad (39)\end{aligned}$$

After the differentiation implied by Equation (21) has been performed, the J matrix for $(v_{jk}, \Phi_{jk})_j^{n+1}$ is

$$J = \begin{bmatrix} D_1 & H_1 & & & \\ -H_2 & D_2 & H_2 & & 0 \\ & \ddots & \ddots & \ddots & \\ 0 & & \ddots & \ddots & H_{N_y-1} \\ & & & -H_{N_y} & D_{N_y} \end{bmatrix}, \qquad (40)$$

where

$$D_k = \begin{bmatrix} 1 + \frac{\lambda_y}{4}(v_{k+1}^{n+1} - v_{k-1}^{n+1}) & \frac{\lambda_y}{8}(\Phi_{k+1}^{n+1} - \Phi_{k-1}^{n+1}) \\ \lambda_y 4(\Phi_{k+1}^{n+1} - \Phi_{k-1}^{n+1}) & 1 + \frac{\lambda_y}{8}(v_{k+1}^{n+1} - v_{k-1}^{n+1}) \end{bmatrix} \quad (\text{fixed } j), \qquad (41)$$

$$H_k = \frac{\lambda_y}{8}\begin{bmatrix} 2v_k^{n+1} & \Phi_k^{n+1} \\ \Phi_k^{n+1} & 2v_k^{n+1} \end{bmatrix} (\text{fixed } j)\,.$$

Note that, in the 2 × 2 matrices H_{N_y} and D_{N_y}, $v_{N_y}^{n+1} = 0$. The *LU* decomposition is performed at every *M*-th time-step, and then $J^{-1}g$ is solved by backsubstitution.

(4) Having solved $(v_{jk}^{n+1}, \Phi_{jk}^{n+1})$ by using the QN method, we solve u_{jk}^{n+1}. The corresponding equation for u_{jk}^{n+1} is [from Equation (16b)]

$$\begin{aligned} u_{jk}^{n+1} &+ \frac{\Delta t}{2} v_{jk}^{n+1}(u_{j,k+1}^{n+1} - u_{j,k-1}^{n+1})/(2\,\Delta y) - \frac{\Delta t}{2} f_k v_{jk}^{n+1} \\ &= u_{jk}^{n+(1/2)} - \frac{\Delta t}{2} u_{jk}^{n+(1/2)}(u_{j+1,k}^{n+(1/2)} - u_{j-1,k}^{n+(1/2)})/(2\,\Delta x) \\ &\quad - \frac{\Delta t}{2}\frac{\Phi_{jk}^{n+(1/2)}}{2}(\Phi_{j+1,k}^{n+(1/2)} - \Phi_{j-1,k}^{n+(1/2)})/(2\,\Delta x)\,. \qquad (42) \end{aligned}$$

Here, we obtain

$$D_k = 1, \qquad H_k = \frac{\lambda_y}{4} v_{jk}^{n+1}, \qquad (43)$$

that is

$$J = \begin{bmatrix} 1 & \frac{\lambda_y}{4} v_1 & & & 0 \\ -\frac{\lambda_y}{4} v_2 & 1 & \frac{\lambda_y}{2} v_2 & & \\ & \ddots & \ddots & \ddots & \\ & & & & \frac{\lambda_y}{4} v_{N_y-1} \\ 0 & & & -\frac{\lambda_y}{4} v_{N_y} & 1 \end{bmatrix} \quad (j, n+1 \text{ fixed})\,.$$

The quasi-Newton method is used again for solving u_{jk}^{n+1}.

THE AUGMENTED LAGRANGIAN METHOD—APPLICATION AND ALGORITHM

Method

We define a function *f*:

$$f = \sum_{f=1}^{N_x} \sum_{k=1}^{N_y} [\tilde{\alpha}(u - \tilde{u})^2 + \tilde{\alpha}(v - \tilde{v})^2 + \tilde{\beta}(h - \tilde{h})^2]_{jk}\,, \qquad (45)$$

where $N_x \Delta x = L$, $N_y \Delta y = D$, and where $\Delta x = \Delta y = h$ is the grid size, n designates the time-level $t_n = n\Delta t$, where Δt is the time step, and L and D are the respective dimensions of the rectangular domain.

$(\tilde{u}, \tilde{v}, \tilde{h})_{jk}^n$ are the predicted variables at the n-th time-step using a finite-difference algorithm (i.e., the non-linear ADI method of Gustafsson, 1971) for solving the nonlinear shallow-water equations, whereas $(u, v, h)_{jk}^n$ are the values adjusted by the nonlinear constrained optimization method using the augmented Lagrangian technique to enforce conservation of the three integral invariants of the shallow-water equations.

Here $\tilde{\alpha}$ and $\tilde{\beta}$ are weights determined by following Sasaki's (1976) principle that the relative weights are so selected as to make the fractional adjustment of variables proportional to the fractional magnitude of the truncation errors in the predicted variables.

In this program we used

$$\tilde{\alpha} = 1, \qquad \tilde{\beta} = g/H\,, \qquad (46)$$

H being the mean-depth of the shallow fluid, and we adopt the same three basic principles as Sasaki (1976). The augmented Lagrangian function L is defined by

$$L(\mathbf{x}, \mathbf{u}, r) = f(x) + u^T e(x) + \frac{1}{2r}|r(x)|^2, \qquad (47)$$

and the minimization of (47) replaces the problem

$$\text{minimize} f(\mathbf{x})\,, \qquad (48)$$

subject to the equality constraints

$$e(\mathbf{x}) = 0, \qquad (49)$$

where

$$\mathbf{x} = (\tilde{u}_{11} \ldots, \tilde{u}_{N_x N_y}, \tilde{v}_{11} \ldots, \tilde{v}_{N_x N_y}, \tilde{h}_{12}, \ldots, \tilde{h}_{N_x N_y})^n \qquad (50)$$

and $e(\mathbf{x})$ is a vector composed of three nonlinear components given by:

$$\mathbf{e}(\mathbf{x}) = \begin{cases} E^n - E^0 \\ Z^n - Z^0\,, \\ H^n - H^0 \end{cases} \qquad (51)$$

where

$$E^n = \frac{1}{2}\sum_{j=1}^{N_x}\sum_{k=1}^{N_y} [\tilde{h}(\tilde{u}^2 + \tilde{v}^2) + g\tilde{h}^2]_{jk}^n \,\Delta x\,\Delta y$$

$$Z^n = \frac{1}{2}\sum_{y=1}^{N_x}\sum_{k=1}^{N_y} \left[\left(\frac{\partial \tilde{v}}{\partial x} - \frac{\partial \tilde{u}}{\partial y} + f\right)/\tilde{h}\right]_{jk}^{2n} \Delta x\,\Delta y$$

$$H^n = \sum_{j=1}^{N_x}\sum_{k=1}^{N_y} \tilde{h}_{jk}\,\Delta x\,\Delta y\,. \qquad (52)$$

Here, E^n, Z^n, and H^n are the discrete values of the integral invariants of total energy, potential enstrophy and mass at time $t_n = n\,\Delta t$, whereas E^0, Z^0, and H^0 are the values of the same integral invariants at the initial time $t = 0$.

In general, if we have m integral invariants, the constraints vector $e(\mathbf{x})$ is given by

$$\mathbf{e}(\mathbf{x}) = (e_1(\mathbf{x}) \ldots e_m(\mathbf{x}))\,. \qquad (53)$$

The vector u is the m-component multiplier vector

$$u = (u_1, u_2 \ldots u_m)\,, \qquad (54)$$

whereas r is a penalty parameter.

The augmented Lagrangian algorithm

Here we follow the algorithm of Bertsekas (1975, 1980) for minimizing the augmented Lagrangian

$$L_{r_k}(\mathbf{x}, u_k) = f(x) + u_k e(\mathbf{x}) + \frac{1}{2r_k}\,|e(\mathbf{x})|^2. \qquad (55)$$

The algorithm proceeds as follows:

First, we either select an initial vector of multipliers u based on a priori knowledge (see Sasaki, 1976; Sasaki, Barker, and Goerss, 1979), or start with a zero vector in the absence of such knowledge. We then select penalty parameters $r_0^i > 0$ and a sequence $\{\eta_k\}$ with $\eta_0 > 0$.

Step 1: Given a multiplier vector u_k, penalty parameters r_k^i and a parameter y_k, locate a vector $\mathbf{x}_k$ satisfying

$$\|\nabla_k L_{r_k}(\mathbf{x}_k, \mathbf{u}_k)\| \le \{\eta_k\}\,\|e(\mathbf{x}_k)\|\,, \qquad (56)$$

by carrying out an inexact unconstrained minimization of the augmented Lagrangian function $L_{r_k}(\mathbf{x}_k, u_k)$.

For the unconstrained minimization we used the conjugate gradient method, which has the virtue of requiring relatively few memory storage locations because we have a large-scale minimization problem. In our situation we used the ZXCGR IMSL routine (see also Navon and de Villiers, 1983).

Step 2: If

$$|e(x_k)| < \varepsilon_i\,, \qquad (57)$$

where ε_i belongs to a preselected decreasing sequence $\{\varepsilon_k\}$ tending to zero, then stop. Otherwise proceed to Step 3.

Step 3: Update the multiplier vector u_k by using the formula

$$u_{k+1} = u_k + r_k^{-1}\mathbf{e}(\mathbf{x}_k)\,. \qquad (58)$$

Update the select penalty parameters $r_{k+1}^i \varepsilon (0, r_k)$, following the formula

$$r = \begin{cases} \beta r_k, & \text{if } |e[x(u_k, r_k)]| > \gamma |e[x(u_{k-1}, r_{k-1})]| \\ r_k, & \text{if } |e[x(u_k, r_k)]| \le \gamma |e[x(u_{k-1}, r_{k-1})]| \end{cases} \qquad (59)$$

where

$$\beta = (0.4)^k$$
$$\gamma = 0.25\,. \qquad (60)$$

Select $\eta_{k+1} \ge 0$, following a formula of the form

$$\eta_k = (0.8)^k. \qquad (61)$$

Return to Step 1 and perform another cycle of augmented Lagrangian minimization.

Formulae to calculate the value of the function (i.e. the augmented Lagrangian) and its gradient are user-supplied and will be described later in connection with subroutine FUNCT.

PROGRAM GUSTAF

The Test Problem

The test problem used here is the same as that in Navon and Riphagen (1979), which is the initial height-field condition No. 1 of Grammelvedt (1969), that is

$$h(x,y) = H_0 + H_1 \tanh\left(9\left(\frac{D/2 - y}{2D}\right)\right) + H_2 \operatorname{sech}^2\left(9\left(\frac{D/2 - y}{D}\right)\right) \cdot \sin\left(\frac{2\pi x}{L}\right). \qquad (62)$$

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

$$u = \left(\frac{-g}{f}\right)\frac{\partial h}{\partial y}\,, \qquad v = \left(\frac{g}{f}\right)\frac{\partial h}{\partial x}\,. \qquad (63)$$

The constants used were:

$$\begin{array}{ll} L = 4400 \text{ km} & g = 10 \text{ m s}^{-2} \\ D = 6000 \text{ km} & H_0 = 2000 \text{ m} \\ \hat{f} = 10^{-4} \text{ s}^{-1} & H_1 = 220 \text{ mm} \\ \beta = 1.5.10^{11} \text{ s}^{-1}\text{m}^{-1} & H_2 = 133 \text{ m}\,. \end{array} \qquad (64)$$

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

$$\begin{array}{ll} \Delta x = \Delta y = 200 \text{ km} & \Delta t = 1800 \text{ s} \\ \Delta x = \Delta y = 200 \text{ km} & \Delta t = 3600 \text{ s}\,. \end{array} \qquad (65)$$

with $M = 6$ or $M = 12$.

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

$$\Delta x = \Delta y = 500 \text{ km}, \qquad \Delta t = 3600 \text{ s}, \quad \text{with } M = 12\,. \qquad (66)$$

The Dissipation Term

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

$$\varepsilon \Delta t^3 D_{+y} D_{-y} w^n_{jk} \quad (67)$$

was added to the right-hand side of equation (16a) and the term

$$\varepsilon \Delta t^3 D_{+x} D_{-x} w^{n+(1/2)}_{jk} \quad (68)$$

to the right-hand side of equation (16b).

The coefficient $\varepsilon = 0.015$ was used.

Program Operations

Input specifications We first shall describe the input specifications and only then the various subroutines of the program GUSTAF. The input to the program consists of two cards, as follows:

CARD 1 FORMAT (6F10.4,3I5,F5.3) contains the following ten parameters:
FL — the length dimension (*L*) of the rectangular integration domain;
D — the width dimension (*D*) of the rectangular integration 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 field should be displayed;
M — the number of time-steps between successive updates of the *LU* decomposition of the Jacobian matrix *J*, for the *QN* method;
NINT — the number of nonlinear *QN* iterations to be performed at each time-step;
ADJ — the value of ε, the diffusion coefficient.

CARD 2 (called in subroutine (SETUP) specifies different parameters relative to the initial field [see Eq. (1)], using format 6E10.4, and contains the following five parameters:
*H*0 — constant for the initial height field;
*H*1 — constant for the initial height field;
*H*2 — constant for the initial height field;
FHAT — Coriolis parameter;
BETA — df/dy, the Rossby parameter.

Main program and subroutines The main program SHALLOW reads the first data card and, after some preliminary calculations, calls the subroutine SETUP to compute the initial height-field and velocity-field values at each grid point.

The output subroutines UVOUT, LOOK, and HOUT (called from LOOK) are next called to display the initial fields, the initial total energy and the initial mean height. Subroutine MAPPA (called from LOOK) displays a printer-plotted map of the height-field contours.

The solution of the nonlinear constrained optimization problem requires scaling of the variables so that the scaled variables are of similar magnitude and of order unity in the region of interest. Also, the nonlinear equality constraints should be of the same order of magnitude to avoid one constraint dominating the others.

This scaling is performed in the main program SHALLOW. The variables are scaled as follows:

$$\begin{aligned} &u^s_{ij} = u_{ij}V^{-1}, \quad v^s_{ij} = v_{ij}V^{-1}, \quad h^s_{ij} = h_{ij}L^{-1} \\ &f_j = Tf_j \quad g^s = gLV^{-2} \\ &i = 1 \ldots N_x \\ &j = 1 \ldots N_y \end{aligned} \quad (69)$$

(see also Navon and de Villiers, 1983; Gill, Murray, and Wright 1981).

After the dissipation term has been calculated, the central subroutine GUSTAF is called. This subroutine performs the bulk of the Gustafsson nonlinear ADI algorithm and solves the values of U, V, PHI for each successive time-step.

GUSTAF in turn calls, at each time-step, the subroutines BACKBLK and BACKTRI to perform block or scalar backsubstitutions, respectively, whereas the subroutines LUDECOM and LUTRID are called every *M*-th time-step to perform block or scalar matrix LU decompositions, respectively.

After a predetermined number of time-steps (IPR), the subroutine LOOK is called to calculate the integral invariants of the shallow-water equations, namely the total energy and the mean height, as well as the potential enstrophy.

Subroutine LOOK in turn calls the output subroutines HOUT and MAPPA. When the preset total simulation time has been reached, the height and velocity fields are written on file for further use, together with the name of the program, the number (*M*) of nonlinear QN iterations per time-step, the number of days of simulation, the time-step, the space increments *DX* and *DY*, and the number of grid points (*NX* and *NY*) in the *x* and *y* directions, respectively.

SUBROUTINE GUSTAF (*U,V,PH,I,H,F,UH, VH,PHIM,AK,BK,CK,DK,EK,AJ,BJ,CJ,P,Q,R, X,Y,Z,G,S,E,AL,BL,CL,DL,EL,AM,BM,CM,LX, NX,NY,NT,M,NINT*).

This subroutine performs the bulk of the work when the Gustafsson nonlinear ADI algorithm is used to solve the shallow-water equations. Essentially, the subroutine follows the same procedure as the algorithm outlined in the section on Implementation of the QN Method in the Gustafsson ADI algorithm.

Starting with the initial fields, *U, V,* and *PHI*, GUSTAF first calls the subroutine LUDECOM every *M*-th time-step to perform cyclic block-tridiagonal LU decomposition. *P, Q,* and *R* are the subdiagonal, diagonal, and superdiagonal (2 × 2) block matrices, respectively, that constitute the Jacobian matrix. In this way we obtain $u^{n+(1/2)}_{jk}$ and $\Phi^{n+(1/2)}_{jk}$, renamed *UH* and *PHIH*, respectively, following Equations (28)–(33), and by calling subroutine BACKBLK to perform the block backsubstitution, we obtain $J^{-1}g$ for the first one-half of the time-step for $(u^{n+(1/2)}_{jk}, \Phi^{n+(1/2)}_{jk})$. The procedure is repeated NINT times, which is the number of preset nonlinear QN iterations for every one-half of a time-step. Then $v^{n+(1/2)}_{jk}$, renamed *VH*, is obtained, following the algorithm outlined in Equations (34)–(36), that is by calling the subroutine BACKTRI to perform cyclic tridiagonal matrix backsubstitution. The procedure again is repeated NINT times — the number of present nonlinear QN iterations for every one-half of a time-step.

The rest of the parameters in GUSTAF pertain to the specific subroutines which it calls, and will be detailed there.

The second part of subroutine GUSTAF, the augmented Lagrangian method for enforcing a posteriori conservation of the shallow-water integram invariants, is implemented.

One first tests whether one needs to carry out an adjustment at a given time-step by measuring the deviation from conservation. Next, one sets up an augmented Lagrangian function, with initial multipliers *UUH*, *UZ*, and *UE* corresponding to the constraints of total mass, potential enstrophy, and total energy, respectively. Then, the initial penalties *PNLTH*, *PNLTZ*, and *PNLTE* are set. An initial value for the parameter *ETA* also is set.

Then a loop is set up which implements the augmented Lagrangian algorithm. A conjugate-gradient unconstrained minimization solver—in this instance the IMSL Library Subroutine ZXCGR using a method due to Powell (1977)—is used to minimize the augmented Lagrangian. The unconstrained minimization is considered to be accomplished once a threshold accuracy dependent on ETA is reached.

Thereafter, the Lagrange multipliers, the penalties and the parameter ETA are updated, and another cycle of augmented Lagrangian minimization is completed. The process will stop either when the nonlinear equality constraints are satisfied within a required accuracy, or when ETA becomes too small, that is, when the number of augmented Lagrangian minimization cycles exceeds a limit.

Subroutine ZXCGR calls the subroutine FUNCT, which evaluates the function value of the augmented Lagrangian, as well as its gradient vector.

If the user has a NAG scientific library, he may select to use subroutines EO4DBF or EO4DBE, double- or single-precision subroutines of the NAG library, which determine an unconstrained minimum of a function of several variables, using first derivatives, by the conjugate gradient algorithm.

SUBROUTINE LUDECOM (*P,Q,R,A,B,C,D,E, NY,N,IND*).

This subroutine performs *LU* decomposition of a block tridiagonal matrix with subdiagonal, diagonal, and superdiagonal elements in arrays *P*, *Q*, and *R*, respectively. The elements of the matrix are (2×2) matrices. The matrix is decomposed into matrices *L* and *U*, where *L* has subdiagonal elements in array *B* and diagonal elements in array *A*, *U* has superdiagonal elements in array *C*, and all diagonal matrices are identity matrices.

If the given block tridiagonal matrix is not cyclic, set *IND* = 0. If the given block tridiagonal matrix is cyclic, set *IND* = 1. In this situation *L* also has nonzero elements in blocks 1 to (*NM*-2) of the last block-row in array *E*, and *U* also has nonzero elements in blocks 1 to (*NM*-2) of the last block column in array *D*.

The order of the given matrix is $2 \times N$.

For the algorithm used for the *LU* decomposition of a cyclic block tridiagonal matrix see Navon (1977).

SUBROUTINE LUTRID (*P,Q,R,A,B,C,D,E, L,N,IND*).

This subroutine performs the *LU* decomposition of a tridiagonal matrix, with subdiagonal, diagonal, and superdiagonal elements in arrays *P*, *Q*, and *R*, respectively.

This matrix is of order *N*.

If the matrix is cyclic tridiagonal, *IND* is set to 1, otherwise to 0. In the *L*-matrix the diagonal elements are in array *A* and the subdiagonal elements in array *B*. If *IND* = 1, the first *N*-2 elements of the *N*-th row are in array.

In the *U*-matrix the diagonal elements are all equal to 1 (therefore not stored) and the superdiagonal elements are in array *C*. If *IND* = 1, the first *N*-2 elements of column *N* are in array *D*.

SUBROUTINE BACKBLK (*A,B,C,D,E,S,X, L,N,IND*).

Following Equations (24)–(25), this subroutine solves $J^{-1}g$ in two stages by backsubstitution, that is, it determines *X* where

$$L * Y = S \quad \text{and} \quad U * X = Y,$$

the matrices *L* and *U* being formed by *LU* decomposition of a block or cyclic block tridiagonal matrix.

Here *X* stands for the vector $J^{-1}g$ in the nonlinear quasi-Newton iterative method.

SUBROUTINE BACKTRI (*A,B,C,D,E,S,X,L, N,IND*)

Following Equations (24)–(25), this subroutine solves $J^{-1}g$ in two stages by backsubstitution, that is determines *X* where

$$L * Y = S \quad \text{and} \quad U * X = Y,$$

the matrices *L* and *U* being formed by *LU* decomposition of a tridiagonal or cyclic tridiagonal matrix.

Here *X* stands for the vector $J^{-1}g$ in the nonlinear quasi-Newton iterative method.

SUBROUTINE SETUP (*U,V,PHI,H,F,NX,NY, S,C,LX*)

This subroutine sets up the initial height field *H* and calculates the variable $PHI = \Phi = 2\sqrt{gh}$ and from it, using Equation (46), the components of the initial velocity fields *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 potential enstrophy, total energy and mean height, which are invariants of the shallow-water equations. It also prints out these values, together with the height-field values, by calling subroutine HOUT, and calls subroutine MAPPA for a lineprinter contour plot of the height field. The CPU time for each 12 time-steps also is printed.

SUBROUTINE MAPPA (*FUN,C,NX,NZ,LX*)

This subroutine provides a visual display of the height field by lineprinting an isoline contour plot of the height field for every fifty meters. The parameter FUN gives the forecast field to be contoured, whereas the parameter C is the inverse of the contour interval in meters (for example, if the contour interval is 50 m, $C = 0.02$). The parameter $NZ = NY + 1$.

SUBROUTINE HOUT (H, NX, NY, LX)

This subroutine digitally prints the height-field values in a matrix format.

SUBROUTINE 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 FUNCT (XC, FC, GC)

This subroutine evaluates the function value of the augmented Lagrangian, as well as the gradient vector of the augmented Lagrangian.

Here XC stands for the length of the vector x (i.e. $3N_xN_y$), whereas FC is the function value and GC the gradient vector value, both of dimension $3N_xN_y$.

Examples of output Examples of GUSTAF output are provided to demonstrate the different options of the program. The initial height field, using a space resolution of $\Delta x = \Delta y = 200$ km, is shown in Figure 1. Figures 2 to 4 show the height-field contours after two simulation days, using a time-step of 3600 s and the methods $QNEX1$, $QN2$, and $QN3$ with $M = 6$. Figures 5 and 6 show the height-field contours, using a space resolution of $\Delta x = \Delta y = 500$ km and a time-step of 3600 s for the methods $QN2$ and $QN3$ with $M = 12$, the dissipation coefficient being $\varepsilon = 0.015$, after 20 simulation days.

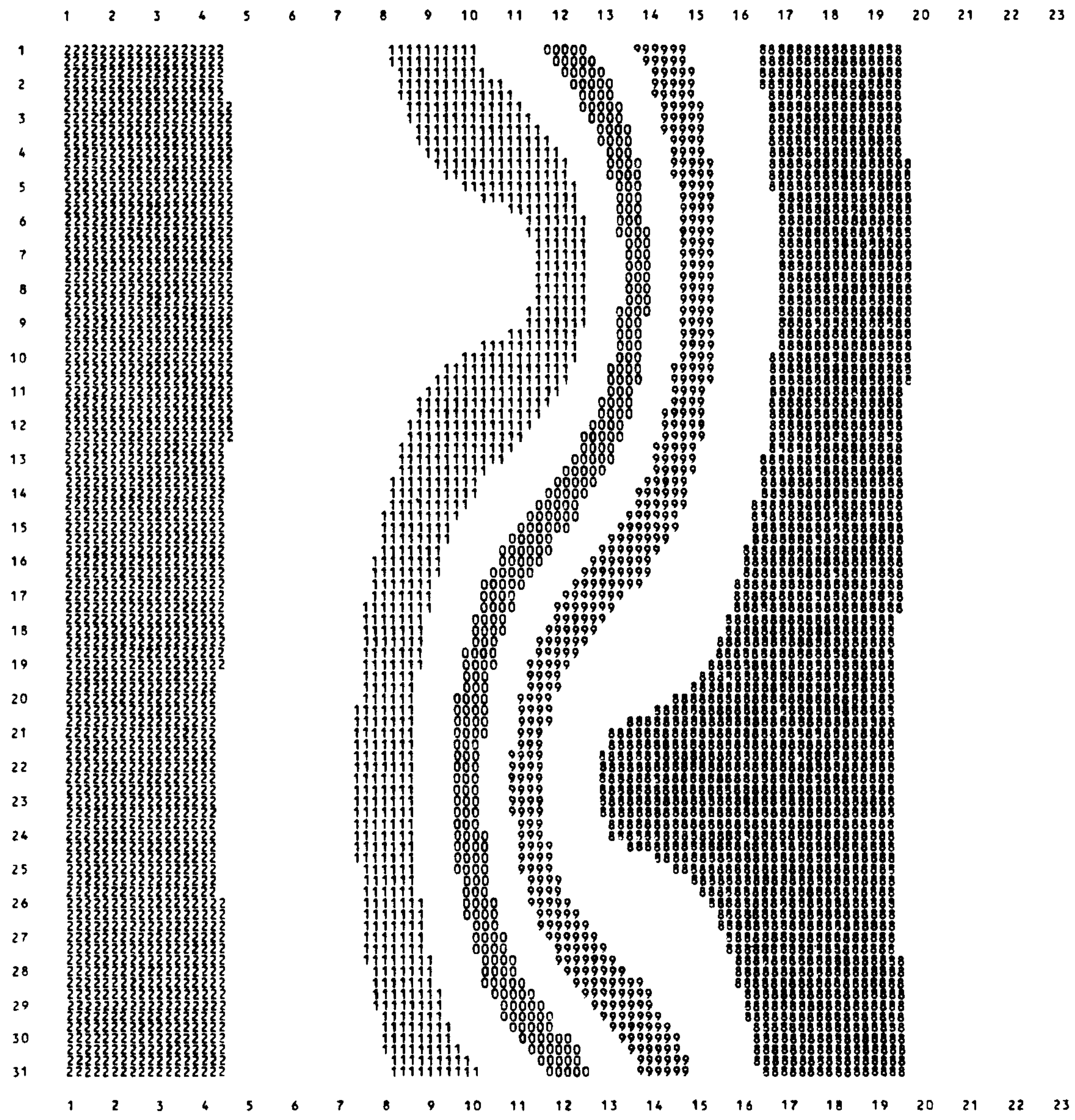


Figure 1. Initial height field. $DX = DY = 200000$ M.

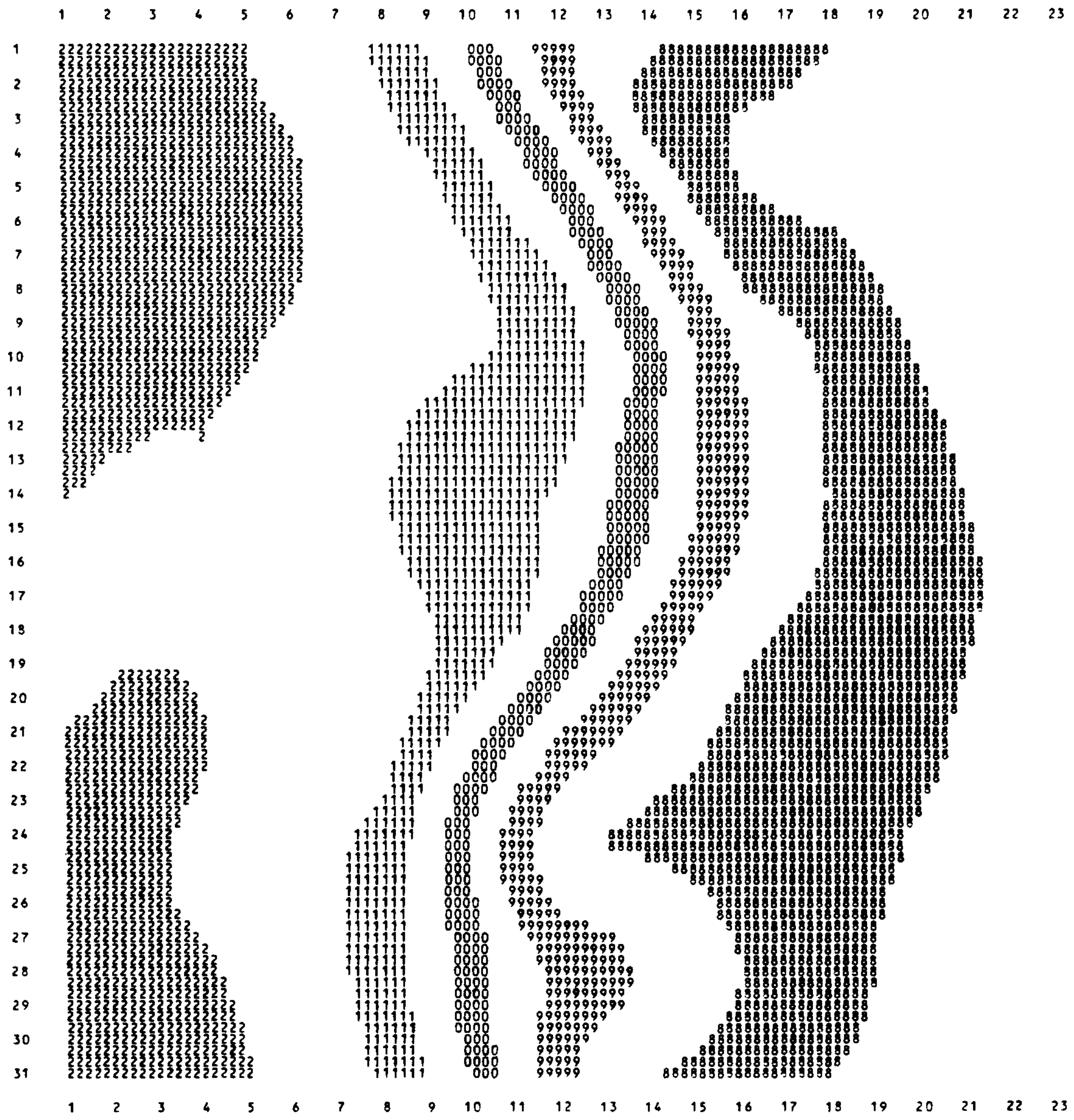


Figure 2. Height-field contours after 2 days. $DX = DY = 200000$ M, $DT = 3600$ sec, *QNEX*1 method with $M = 6$.

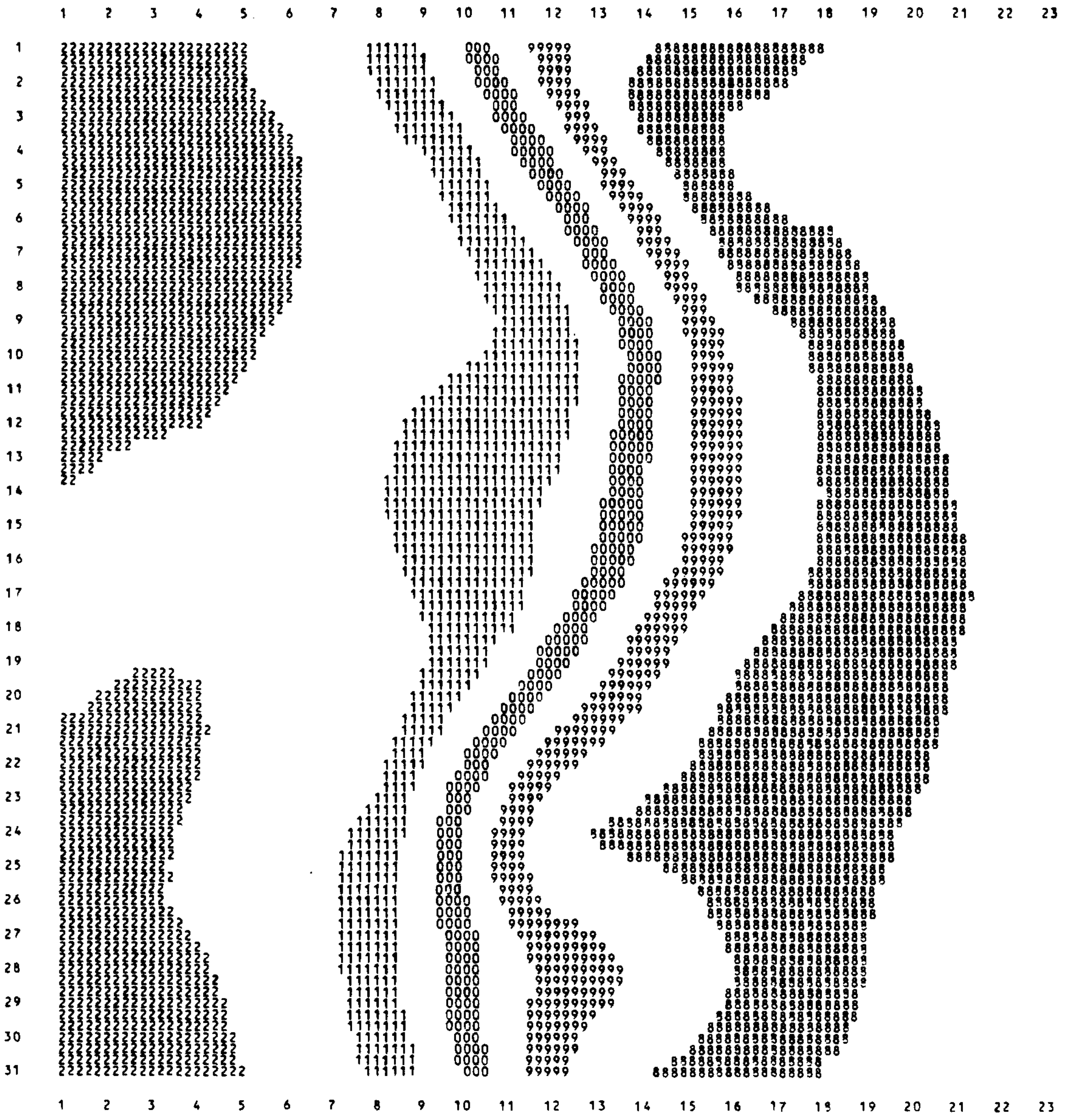


Figure 3. Height-field contours after 2 days. $DX = DY = 200000$ M, $DT = 3600$ sec, $QN2$ method with $M = 6$.

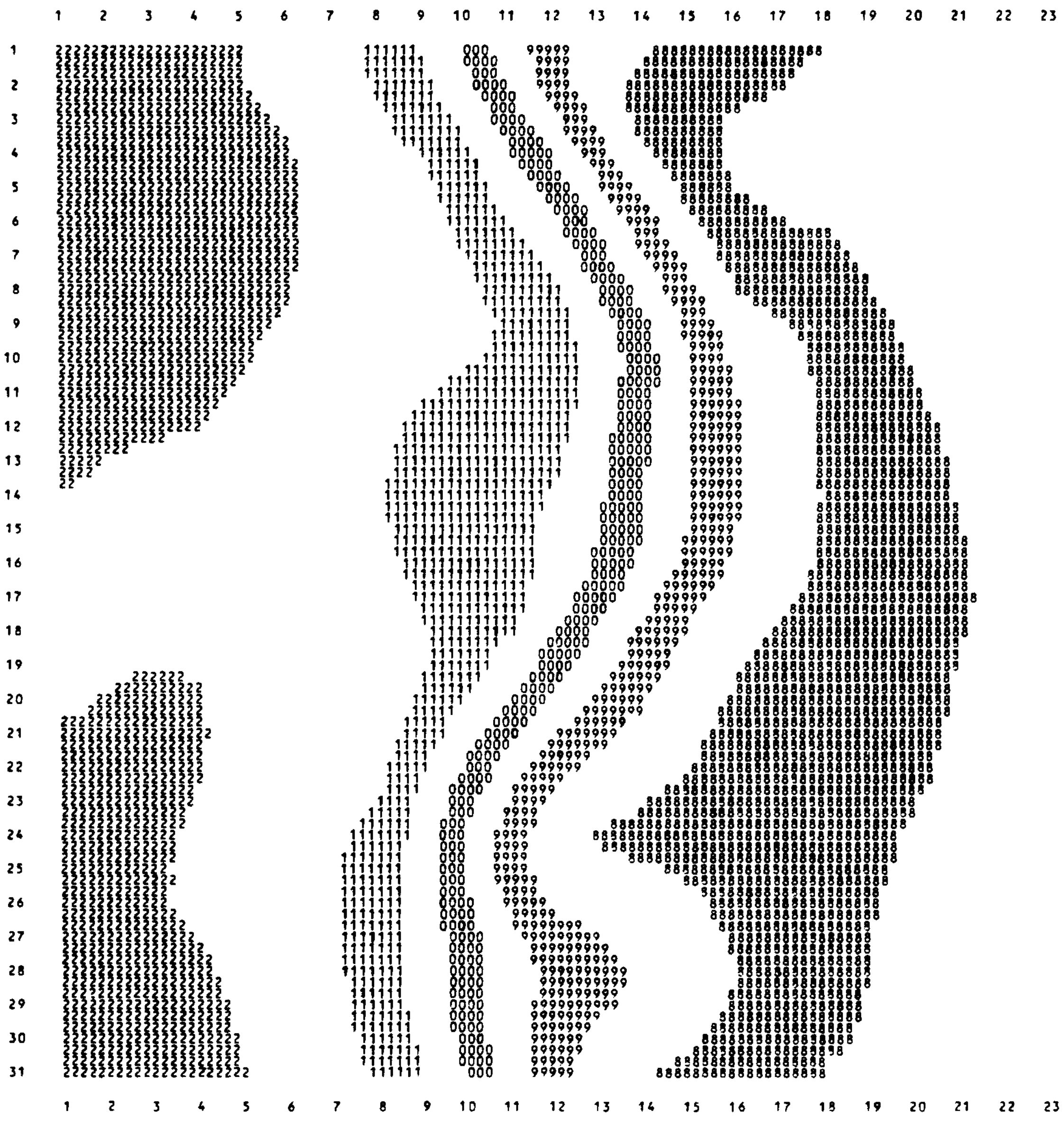


Figure 4. Height-field contours after 2 days. $DX = DY = 200000$ M, $DT = 3600$ sec, $QN3$ method with $M = 6$.

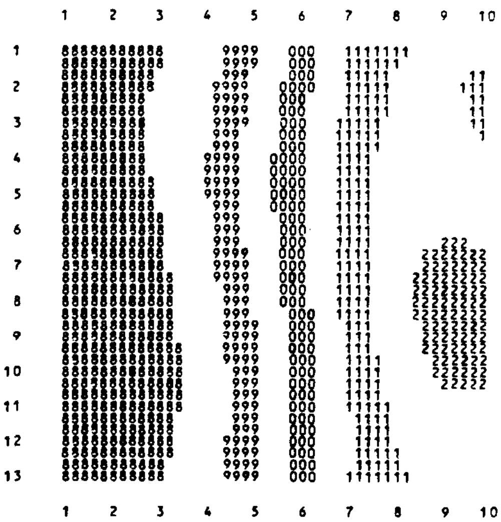


Figure 5. Height-field contours after 20 days. $DX = DY = 500000$ M, $DT = 3600$ sec, $QN2$ method with $M = 12$, epsilon $= 0.015$.

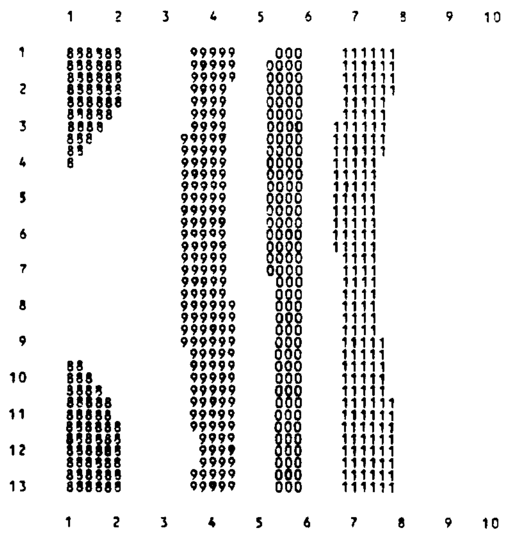


Figure 6. Height-field contours after 20 days. $DX = DY = 500000$ M, $DT = 3600$ sec, $QN3$ method with $M = 12$, epsilon $= 0.015$.

REFERENCES

Bertsekas, D. P., 1975, Combined primal dual and penalty methods for constrained optimization: SIAM Jour. Control and Optimization, v. 13, p. 521–544.

Bertsekas, D. P., 1980, Penalty and multiplier methods in nonlinear optimization: theory and algorithms, *in* Dixon, L. C. W., Spedicato, E., and Szego, C. P., eds., Publishers of Nonlinear optimization: theory and algorithms, Boston, p. 253–278.

Gill, P. E., Murray, W., and Wright, M. H., 1981, Practical optimization: Academic Press, New York, p. 401.

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. Comput. Physics, v. 7, no. 2, p. 239–254.

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, no. 3, p. 141–150.

Isaacson, E., and Keller, H. B., 1966, Analysis of numerical methods: John Wiley & Sons, New York, 541 p.

Kreiss, H. O., and Widlund, O. B., 1966, Difference approximations for initial-value problems for partial differential equations, *in* Kruskal, M. ed., Proceedings of the Summer School for Mathematics and Physics: Springer-Verlag, New York, 301 p.

Navon, I. M., 1977, Algorithms for the solution of scalar and block cyclic tridiagonal systems: CSIR Spec. Rep. WISK 265, Pretoria, South Africa, 45 p.

Navon, I. M., and Riphagen, H. A., 1979, An implicit compact fouth-order algorithm for solving the shallow-water equations in conservation-law form: Mon. Wea. Rev., v. 107, no. 9, p. 1107–1127.

Navon, I. M., and de Villiers, R., 1983, Combined penalty-multiplier optimizaton methods to enforce integral invariants conservation: Mon. Wea. Rev., v. 111, no. 6, p. 1228–1243.

Navon, I. M., and Riphagen, H. A., 1986, SHALL 4—an implicit compact fourth-order FORTRAN program for solving the shallow water equations in conservation-law form: Computers & Geosciences, v. 12, no. 2, this issue.

Powell, M. J., 1977, Restart procedures for the conjugate-gradient method: Math. Program No. 12, p. 241–254.

Sasaki, J., 1976, Variational design of finite-difference schemes for initial-value problems with an integral invariant: Jour. Comput. Physics, v. 21, p. 270–278.

Sasaki, J., Barker, T., and Goerss, J. S., 1979, Dynamic data assimilation by the noise freezing method: Final Report No. F52551792, Naval Environmental Prediction Research Facility, Monterey, California 93940, 80 p.

APPENDIX

The shallow-water equations for the atmosphere

The free surface linearized gravity wave equations for a one-layer, homogeneous incompressible fluid with an upper surface permitted to be free are

$$\frac{\partial u}{\partial t} + U\frac{\partial u}{\partial x} + \frac{1}{\bar{\rho}}\frac{\partial p}{\partial x} = 0 \tag{A1}$$

$$\delta\left(\frac{\partial w}{\partial t} + U\frac{\partial w}{\partial x}\right) + \frac{1}{\bar{\rho}}\frac{\partial p}{\partial z} = 0 \tag{A2}$$

$$\frac{\partial u}{\partial x} + \frac{\partial w}{\partial z} = 0\,. \tag{A3}$$

Here δ identifies terms that would contribute to the divergence equation. Its value is either 0 or 1.

Using the hydrostatic assumption one obtains

$$\frac{1}{\bar{\rho}}\frac{\partial p}{\partial x} = g\frac{\partial h}{\partial x} \tag{A4}$$

and Equation (A1) becomes

$$\frac{\partial u}{\partial t} + U\frac{\partial u}{\partial x} + g\frac{\partial h}{\partial x} = 0 \tag{A5}$$

By integrating the continuity Equation (A3) in the vertical one obtains

$$\frac{\partial h}{\partial t} + U\frac{\partial h}{\partial x} + H\frac{\partial u}{\partial x} = 0\,. \tag{A6}$$

The phase velocity for shallow-water waves is

$$c = U \pm \sqrt{gH}. \tag{A7}$$

The gravity waves described are termed *external*, because their maximum amplitude is at the boundary of the fluid.

If the effects of the earth's rotation are added to the hydrostatic one-layer equations—the deflection caused by the Coriolis force affects low-frequency gravity waves. In addition Rossby waves are determined that depend on the spatial variation of the Coriolis parameter. The equations of motion are

$$\begin{aligned}
&\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} - fv + g\frac{\partial h}{\partial x} = 0 \\
&\frac{\partial v}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial v}{\partial y} + fu + g\frac{\partial h}{\partial y} = 0 \qquad \text{(A8)} \\
&\frac{\mathrm{d}h}{\mathrm{d}t} = -\left(\frac{\partial h}{\partial t} + u\frac{\partial h}{\partial x} + v\frac{\partial h}{\partial y}\right).
\end{aligned}$$

Using assumptions of hydrostaticity, constant density, and incompressibility as well as the barotropic assumption that is the density is a functional of pressure alone, which makes all surface pressure surfaces to be parallel and only one level need be forecast. Using harmonic perturbations one obtains a cubic frequency equation

$$\delta(U - c)^3 - (gH + f^2/k^2)(U - c) - \frac{fg}{k^2}\frac{\partial H}{\partial y} = 0. \tag{A9}$$

To get the fast solutions set $U = 0$ and $\delta = 1$ which gives

$$c_{1,2} = \pm\sqrt{gH + f^2/\mu^2}. \tag{A10}$$

These are inertial gravity-waves and when $f = 0$

$$c_{1,2} = \pm\sqrt{gH} \tag{A11}$$

that is the formula for shallow-water waves.

The slow meteorological solution to (A9) may be obtained by setting $\delta = 0$

$$c_3 = \frac{U + (f/H)\dfrac{\partial H}{\partial y}}{k^2 + (f^2/gH)}. \tag{A12}$$

PROGRAM LISTING

```
C       PROGRAM GUSTAV(INPUT,OUTPUT,TAPE1=INPUT,TAPE3=OUTPUT,TAPE11)
C   THIS IS THE MAIN CONTROL PROGRAM WHICH IMPLEMENTS THE QUASI-NEWTON
C   NON-LINEAR GUSTAFSSON ALTERNATING IMPLICIT METHOD FOR SOLVING THE
C   SHALLOW WATER EQUATIONS.
C
C   THE PROGRAM ALSO IMPLEMENTS A SCALING OF THE VARIABLES IN ORDER TO
C   ALLOW FOR THE CONSTRAINED MINIMIZATION USING THE AUGMENTED LAGRANGIAN
C   TECHNIQUE.
        COMMON/FLD/XC(540)
        COMMON/OLD/XO(540),H0,Z0,E0,ALPHA,BETA,TG,F(12),NX,NY,LX,LY,PNLTH
       1,PNLTZ,PNLTE,UUH,UZ,UE
        COMMON/CONST/FL,D,T,DX,DY,DT,FX,FY,FT,G,TIME,IPR,ADJ,BDJ,IND
        COMMON/RITE/NIN,NOUT,NTAPE
        DIMENSION U(15,12),V(15,12),PHI(15,12),H(15,12),UH(15,12),
       1 VH(15,12),PHIH(15,12),AK(4,15,12),BK(4,15,12),CK(4,15,12),
       2 DK(4,15,12),EK(4,15,12),AJ(4,12,15),BJ(4,12,15),CJ(4,12,15),
       3 P(4,15),Q(4,15),R(4,15),X(15),Y(15),Z(15),GG(15),S(2,15),
       4 AL(15,12),BL(15,12),CL(15,12),DL(15,12),EL(15,12),AM(12,15),
       5 BM(12,15),CM(12,15),E(2,15)
        EQUIVALENCE (U(1),XC(1)),(V(1),XC(181)),(H(1),XC(361))
        G=10.
        DATA UFAC/1.E03/,HFAC/1.E05/
        NIN=1
        NOUT=3
        NTAPE=11
      1 FORMAT(6F10.4,3I5,F5.3)
      2 FORMAT(42H0 THE LU DECOMPOSITION IS DONE ONLY EVERY ,I2,13H-TH TIM
       1E STEP/53H0 THE NUMBER OF ITERATIONS IN EACH HALF TIME STEP IS ,I2
       2/12H0 EPSILON = ,F7.5)
      7 FORMAT(73H0 CHANGE DIMENSIONS OF ARRAYS  U,V,......,W  TO ACCOMMODA
       1TE THIS DATA SET,/5X,81HAND THE VALUES ASSIGNED TO  LX  AND  LY, W
       2HICH INDICATE CERTAIN ARRAY DIMENSIONS.)
      8 FORMAT(20H1     INITIAL U-FIELD)
      9 FORMAT(20H1     INITIAL V-FIELD)
     10 FORMAT(41H0INITIAL VALUE OF CONSTRAINTS, H, Z & E: ,3E14.6)
  111      FORMAT(5X,2I5)
C   FL, D, T  ARE THE MAXIMUM VALUES OF  X, Y, TIME  RESPECTIVELY.
C      0.LE.X.LE.FL ,   0.LE.Y.LE.D ,   0.LE.TIME.LE.T
C   DX, DY, DT  ARE THE INCREMENTS IN  X, Y, TIME  RESPECTIVELY.
C   IPR INDICATES THE PRINTOUT FREQUENCY, PRINTOUT AFTER EACH DT*IPR.
C      IF IPR IS BLANK OR ZERO  IPR WILL BE GIVEN THE VALUE 1.
         READ(NIN,1) FL,D,T,DX,DY,DT,IPR,M,NINT ,ADJ
         LX=15
         LY=12
         NX=FL/DX
         NY=1+IFIX(D/DY)
         WRITE(NOUT,111) NX,NY
         IF(NX.GT.LX) GO TO 45
         IF(NY.LE.LY) GO TO 50
     45  WRITE(NOUT,7)
         GO TO 250
     50  NT=T/DT
 CALL SETUP TO
C    READ CONSTANTS OF HEIGHT FUNCTION AND OF F,
C    SET UP VECTOR  F(K)=FHAT+BETA*(Y(K)-D/2) FOR K=1,...,NY,
C    COMPUTE INITIAL VALUES OF HEIGHT AND OF OMEGA (I.E. H AND U,V,PHI)
C      FOR EACH POINT OF THE GRID,
C    PRINT OUT INITIAL CONSTANTS.
C      ( U )                U=-(G/F)*(DH/DY)
C    W=( V )    WHERE       V= (G/F)*(DH/DX)
C      (PHI)                PHI=2*SQRT(G*H)
C    W(X,Y,TIME) = W(X+FL,Y,TIME)
C    V(X,0,TIME) = V(X,D,TIME) = 0.
```

```
C   DW/DT = A(W)*(DW/DX) + B(W)*(DW/DY) +C(W)*W,
C                  ( U     0 PHI/2)         ( V     0       0  )       ( 0   F   0 )
C     WHERE   A = (  0     U    0  ),   B = ( 0     V    PHI/2),   C = (-F   0   0 )
C                 (PHI/2  0     U  )        ( 0  PHI/2     V   )       ( 0   0   0 )
      CALL SETUP(U,V,PHI,H,F,NX,NY,AK,BK,LX,LY)
      WRITE(NOUT,2) M,NINT,ADJ
C    PRINT OUT INITIAL VALUES OF  U, V, H, ENERGY, MEAN HEIGHT, ELAPSED
C     TIME, AND HEIGHT CONTOURS.
      WRITE(NOUT,8)
      CALL UVOUT(U,NX,NY,LX)
      WRITE(NOUT,9)
      CALL UVOUT(V,NX,NY,LX)
      TIME=0.
C       SCALE VARIABLES
      DO 55 J=1,NY
      F(J)=F(J)*HFAC/UFAC
      DO 55 I=1,NX
      U(I,J)=U(I,J)/UFAC
      V(I,J)=V(I,J)/UFAC
      PHI(I,J)=PHI(I,J)/UFAC
   55 H(I,J)=H(I,J)/HFAC
      G=G*HFAC/UFAC**2
      DX=DX/HFAC
      DY=DY/HFAC
      DT=DT*UFAC/HFAC
      ADJ=ADJ*HFAC/UFAC**3
      FX=DT/(4.*DX)
      FY=DT/(4.*DY)
      FT=0.5*DT
      CALL LOOK(U,V,PHI,H,NX,NY,LX,F,LY,H0,Z0,E0)
      CALL HOUT(H,NX,NY,LX)
      CALL MAPPA(H,2.E03,NX,NY,LX)
      WRITE(NOUT,10) H0,Z0,E0
      IF(IPR.EQ.0) IPR=1
      EPSDT=ADJ*DT*DT*DT
      ADJ=EPSDT/(DX*DX)
      BDJ=EPSDT/(DY*DY)
C    SOLVE FOR VALUES OF U, V, PHI  FOR EACH SUCCESSIVE TIME STEP.
      CALL GUSTAF(PHI,UH,VH,PHIH,AK,BK,CK,DK,EK,AJ,BJ,CJ,P,Q,R,
     1 X,Y,Z,GG,S,E,AL,BL,CL,DL,EL,AM,BM,CM,NT,M,NINT)
  250 STOP
      END
      SUBROUTINE LUDCOM(P,Q,R,A,B,C,D,E,NY,N,IND)
C   THIS SUBROUTINE PERFORMS A LU (LOWER-UPPER) DECOMPOSITION FOR EITHER
C   A BLOCK-TRIDIAGONAL OR A CYCLIC BLOCK-TRIDIAGONAL MATRIX.
C
C   GIVEN A BLOCK (2*2) TRIDIAGONAL MATRIX WITH SUB-DIAGONAL, DIAGONAL,
C     AND SUPER-DIAGONAL ELEMENTS IN ARRAYS  P, Q, R, RESPECTIVELY,
C    TO DECOMPOSE INTO MATRICES  L, AND  U,  WHERE
C       L HAS SUB-DIAGONAL ELEMENTS IN ARRAY B,
C         AND      DIAGONAL ELEMENTS IN ARRAY A,   AND
C       U HAS SUPER-DIAGONAL ELEMENTS IN ARRAY C,
C         AND ALL DIAGONAL MATRICES ARE IDENTITY MATRICES.
C   IF THE GIVEN MATRIX IS NOT CYCLIC, SET IND=0.
C   IF THE GIVEN MATRIX IS CYCLIC, SET IND=1.  IN THIS CASE
C       L HAS ELEMENTS OF BLOCKS 1 TO (NM-2) OF THE LAST BLOCK-ROW IN
C         ARRAY E, AND
C       U HAS ELEMENTS OF BLOCKS 1 TO (NM-2) OF THE LAST BLOCK-COLUMN IN
C         ARRAY D.
C   THE ORDER OF THE GIVEN MATRIX IS  2*N.
C    P(1)=A(1)*D(1)
C    P(I)=B(I) FOR I=2 TO N-1
C    P(N)=B(N)+E(N-2)*C(N-2)
C    R(I)=A(I)*C(I) FOR I=1 TO N-2
C    R(N-1)=B(N-1)*D(N-2)+A(N-1)*C(N-1)
C    R(N)=E(1)
C    Q(1)=A(1)
C    Q(I)=B(I)*C(I-1)+A(I)   FOR I=2 TO N-1
C    Q(N)=SUM(E(1)*D(1)+...+E(N-2)*D(N-2))+B(N)*C(N-1)+A(N)
C    E(I)*C(I)+E(I+1)=0. FOR I=1 TO N-2
      COMMON/RITE/NIN,NOUT,NTAPE
      DIMENSION P(4,NY),Q(4,NY),R(4,NY),A(4,NY),B(4,NY),C(4,NY),D(4,NY),
     1 E(4,NY)
      NM=N-1
      DO 60 I=1,N
      IF(I.GT.1) GO TO 20
      DO 15 L=1,4
   15 A(L,1)=Q(L,1)
      GO TO 30
   20 IM=I-1
C        B(I) = P(I)
      DO 25 L=1,4
   25 B(L,I)=P(L,I)
      IF(I.NE.N) GO TO 28
      IF(IND.EQ.0) GO TO 28
C        B(N) = P(N)-E(N-2)*C(N-2)
      DO 27 L=1,4
   27 B(L,I)=B(L,I)+E(L,NM)
C        A(I) = Q(I) - B(I)*C(I-1)
   28 A(1,I)=Q(1,I)-B(1,I)*C(1,IM)-B(3,I)*C(2,IM)
      A(2,I)=Q(2,I)-B(2,I)*C(1,IM)-B(4,I)*C(2,IM)
      A(3,I)=Q(3,I)-B(1,I)*C(3,IM)-B(3,I)*C(4,IM)
      A(4,I)=Q(4,I)-B(2,I)*C(3,IM)-B(4,I)*C(4,IM)
      IF(I.EQ.N) GO TO 50
C        C(I) = A(I)**(-1)*R(I)
   30 DETI=A(1,I)*A(4,I)-A(2,I)*A(3,I)
      IF(ABS(DETI).GT.1.E-100) GO TO 31
      WRITE(NOUT,1) I,((P(L,K),L=1,4),(Q(L,K),L=1,4),(R(L,K),L=1,4),K=1,
     *N)
    1 FORMAT(1H0,I5/(1P12E11.3))
      WRITE(NOUT,1) I,((A(L,K),L=1,4),(B(L,K),L=1,4),(C(L,K),L=1,4),K=1,
     *I)
      STOP
   31 DETI=1./DETI
      T1=R(1,I)
      T2=R(2,I)
      T3=R(3,I)
      T4=R(4,I)
      IF(I.LT.NM) GO TO 33
      IF(IND.EQ.0) GO TO 33
      T1=T1-B(1,I)*D(1,IM)-B(3,I)*D(2,IM)
      T2=T2-B(2,I)*D(1,IM)-B(4,I)*D(2,IM)
      T3=T3-B(1,I)*D(3,IM)-B(3,I)*D(4,IM)
      T4=T4-B(2,I)*D(3,IM)-B(4,I)*D(4,IM)
   33 C(1,I)= (A(4,I)*T1-A(3,I)*T2)*DETI
      C(2,I)=-(A(2,I)*T1-A(1,I)*T2)*DETI
      C(3,I)= (A(4,I)*T3-A(3,I)*T4)*DETI
      C(4,I)=-(A(2,I)*T3-A(1,I)*T4)*DETI
      IF(IND.EQ.0) GO TO 60
      IF(I.GT.1) GO TO 40
      S1=0.
      S2=0.
      S3=0.
      S4=0.
```

```
C        E(1) = R(N)
      DO 35 L=1,4
   35 E(L,1)=R(L,N)
      T1=-P(1,I)
      T2=-P(2,I)
      T3=-P(3,I)
      T4=-P(4,I)
      GO TO 45
C        E(I) =-E(I-1)*C(I-1)
   40 E(1,I)=-(E(1,IM)*C(1,IM)+E(3,IM)*C(2,IM))
      E(2,I)=-(E(2,IM)*C(1,IM)+E(4,IM)*C(2,IM))
      E(3,I)=-(E(1,IM)*C(3,IM)+E(3,IM)*C(4,IM))
      E(4,I)=-(E(2,IM)*C(3,IM)+E(4,IM)*C(4,IM))
      IF(I.EQ.NM) GO TO 60
      T1=B(1,I)*D(1,IM)+B(3,I)*D(2,IM)
      T2=B(2,I)*D(1,IM)+B(4,I)*D(2,IM)
      T3=B(1,I)*D(3,IM)+B(3,I)*D(4,IM)
      T4=B(2,I)*D(3,IM)+B(4,I)*D(4,IM)
C        D(I) =-A(I)**(-1)*(B(I)*D(I-1))
   45 D(1,I)=-(A(4,I)*T1-A(3,I)*T2)*DETI
      D(2,I)= (A(2,I)*T1-A(1,I)*T2)*DETI
      D(3,I)=-(A(4,I)*T3-A(3,I)*T4)*DETI
      D(4,I)= (A(2,I)*T3-A(1,I)*T4)*DETI
      S1=S1+E(1,I)*D(1,I)+E(3,I)*D(2,I)
      S2=S2+E(2,I)*D(1,I)+E(4,I)*D(2,I)
      S3=S3+E(1,I)*D(3,I)+E(3,I)*D(4,I)
      S4=S4+E(2,I)*D(3,I)+E(4,I)*D(4,I)
      GO TO 60
   50 IF(IND.EQ.0) GO TO 60
C        A(N) = Q(N)-B(N)*C(N-1)- SUM(E(I)*D(I),I=1,(N-2))
      A(1,I)=A(1,I)-S1
      A(2,I)=A(2,I)-S2
      A(3,I)=A(3,I)-S3
      A(4,I)=A(4,I)-S4
   60 CONTINUE
      RETURN
      END
      SUBROUTINE GUSTAF(PHI,UH,VH,PHIH,AK,BK,CK,DK,EK,AJ,BJ,CJ,
     1 P,Q,R,X,Y,Z,G,S,E,AL,BL,CL,DL,EL,AM,BM,CM,NT,M,NINT)
C  THIS SUBROUTINE PERFORMS THE BULK OF THE WORK OF THE GUSTAFSSON (1971)
C  QUASI-NEWTON NONLINEAR ADI ALGORITHM TO SOLVE THE NONLINEAR SHALLOW
C  WATER EQUATIONS.
C  IN ITS SECOND PART IT IMPLEMENTS THE AUGMENTED LAGRANGIAN METHOD FOR
C  'A POSTERIORI' CONSERVATION OF THE SHALLOW WATER INTEGRAL INVARIANTS.
      COMMON/FLD/XC(540)
      COMMON/OLD/XO(540),H0,Z0,E0,ALPHA,BETA,TG,F(12),NX,NY,LX,LY,PNLTH
     1,PNLTZ,PNLTE,UUH,UZ,UE
      COMMON/CONST/FL,D,T,DX,DY,DT,FX,FY,FT,GG,TIME,IPR,ADJ,BDJ,IND
      COMMON/RITE/NIN,NOUT,NTAPE
      DIMENSION U(15,12),V(15,12),PHI(15,12),UH(15,12),VH(15,12),
     1 PHIH(15,12),H(15,12),AK(4,15,12),BK(4,15,12),CK(4,15,12),
     2 DK(4,15,12),EK(4,15,12),AJ(4,12,15),BJ(4,12,15),CJ(4,12,15),
     3 P(4,12),Q(4,12),R(4,12),X(12),Y(12),Z(12),G(12),S(2,12),
     4 AL(15,12),BL(15,12),CL(15,12),DL(15,12),EL(15,12),AM(12,15),
     5 BM(12,15),CM(12,15),E(2,15),FF(12)
      DIMENSION GC(540),WS(3240),
     *HC(482),ZC(482),EC(482),TC(482)
      EXTERNAL FUNCT
      EQUIVALENCE (U(1),XC(1)),(V(1),XC(181)),(H(1),XC(361))
      NN=NX*NY*3
      ALPHA=1.
      BETA=GG/H0
      TG=GG+GG
C  RELATIVE ERROR BOUNDS FOR ACTIVATING THE AUGMENTED LAGRANGIAN ALGORITHM.
      EPSH=H0*5.E-3
      EPSZ=Z0*1.E-3
      EPSE=E0*5.E-3
C  INITIAL PENALTY PARAMETERS.
C  RH0=0.0 INDICATES THAT TOTAL MASS WILL NOT BE USED AS A CONSTRAINT.
      RH0=0.0
      RZ0=0.5
      RE0=0.5
      IFAIL=0
      DO 100 K=1,NY
  100 FF(K)=F(K)*FT
      ADJ=-ADJ
      BDJ=-BDJ
      FX2=0.5*FX
      NYM=NY-1
      KD=M
      IOPT=0
      HC(1)=1.
      ZC(1)=1.
      EC(1)=1.
      TC(1)=0.
      IT1=1
      DO 360 I=1,NT
      IOPT=IOPT+1
      IF(KD.NE.M) GO TO 150
      DO 115 K=1,NY
      DO 110 J=1,NX
      JP1=J+1
      JM1=J-1
      IF(JP1.GT.NX) JP1=1
      IF(JM1.LT.1) JM1=NX
      UJ=(U(JP1,K)-U(JM1,K))*FX
      PJ=(PHI(JP1,K)-PHI(JM1,K))*FX
      Q(1,J)=1.+UJ
      Q(2,J)=PJ
      Q(3,J)=0.5*PJ
      Q(4,J)=1.+0.5*UJ
      R(1,J)=FX*U(J,K)
      R(4,J)=R(1,J)
      R(2,J)=FX2*PHI(J,K)
      R(3,J)=R(2,J)
      DO 105 L=1,4
  105 P(L,J)=-R(L,J)
  110 CONTINUE
      CALL LUDCOM(P,Q,R,AK(1,1,K),BK(1,1,K),CK(1,1,K),DK(1,1,K),
     1 EK(1,1,K),LX,NX,1)
  115 CONTINUE
C  OBTAIN UH, PHIH.
  150 DO 152 K=1,NY
      DO 152 J=1,NX
      UH(J,K)=U(J,K)
  152 PHIH(J,K)=PHI(J,K)
      DO 195 L=1,NINT
      DO 185 K=1,NY
      FKT=FF(K)
      FU=FY
      KP1=K+1
      KM1=K-1
      IF(KP1.LE.NY) GO TO 155
      KP1=NY
      GO TO 160
  155 IF(KM1.GE.1) GO TO 165
      KM1=1
  160 FU=FU+FU
  165 FP=0.5*FU
      DO 170 J=1,NX
      JP1=J+1
      JM1=J-1
      IF(JP1.GT.NX) JP1=1
      IF(JM1.LT.1) JM1=NX
      S(1,J)=U (J,K)-V(J,K)*(FU*(U (J,KP1)-U (J,KM1))-FKT)
      S(2,J)=PHI (J,K)*(1.-FP*(V (J,KP1)-V (J,KM1)))
     1 -FU*V (J,K)*(PHI (J,KP1)-PHI (J,KM1))
      S(1,J)=UH(J,K)*(1.+FX*(UH(JP1,K)-UH(JM1,K)))+FX2*PHIH(J,K)*
     1 (PHIH(JP1,K)-PHIH(JM1,K))-S(1,J)
      S(2,J)=PHIH(J,K)*(1.+FX2*(UH(JP1,K)-UH(JM1,K)))+FX* UH(J,K)*
     1 (PHIH(JP1,K)-PHIH(JM1,K))-S(2,J)
      IF(ADJ.EQ.0.) GO TO 170
      IF(K.EQ.1.OR.K.EQ.NY) GO TO170
      S(1,J)=S(1,J)+ADJ*(U(J,KP1)-2.*U(J,K)+U(J,KM1))
      S(2,J)=S(2,J)+ADJ*(PHI(J,KP1)-2.*PHI(J,K)+PHI(J,KM1))
  170 CONTINUE
      CALL BAKBLK(AK(1,1,K),BK(1,1,K),CK(1,1,K),DK(1,1,K),EK(1,1,K),
     1 S,E,LX,NX,1)
  175 DO 180 J=1,NX
      UH(J,K)=UH(J,K)-E(1,J)
  180 PHIH(J,K)=PHIH(J,K)-E(2,J)
  185 CONTINUE
  195 CONTINUE
  200 CONTINUE
C   OBTAIN VH.
      IF(KD.NE.M) GO TO 215
      DO 210 K=2,NYM
      DO 205 J=1,NX
      Y(J)=1.
      Z(J)=FX*UH(J,K)
  205 X(J)=-Z(J)
      CALL LUTRID(X,Y,Z,AL(1,K),BL(1,K),CL(1,K),DL(1,K),EL(1,K),LX,NX,1)
  210 CONTINUE
  215 DO 220 K=1,NY
      DO 220 J=1,NX
  220 VH(J,K)=V(J,K)
      DO 265 L=1,NINT
      DO 250 K=2,NYM
      FKT=FF(K)
      FU=FY
      KP1=K+1
      KM1=K-1
  230 FP=0.5*FU
      DO 235 J=1,NX
      JP1=J+1
      JM1=J-1
      IF(JP1.GT.NX) JP1=1
      IF(JM1.LT.1) JM1=NX
      Z(J)=V (J,K)*(1.-FU*(V (J,KP1)-V (J,KM1)))-PHI (J,K)*FP*
     1 (PHI (J,KP1)-PHI (J,KM1))
      Z(J)=VH(J,K)+UH(J,K)*(FX*(VH(JP1,K)-VH(JM1,K))+FKT)-Z(J)
       IF(ADJ.EQ.0.) GO TO 235
      Z(J)=Z(J)+ADJ*(V(J,KP1)-2.*V(J,K)+V(J,KM1))
  235 CONTINUE
      CALL BAKTRI(AL(1,K),BL(1,K),CL(1,K),DL(1,K),EL(1,K),Z,G,LX,NX,1)
      DO 245 J=1,NX
  245 VH(J,K)=VH(J,K)-G(J)
  250 CONTINUE
  260 CONTINUE
      DO 265 J=1,NX
      VH(J,1)=0.
  265 VH(J,NY)=0.
C   OBTAIN V, PHI FOR NEXT STEP.
      IF(KD.NE.M) GO TO 268
      DO 145 J=1,NX
      DO 140 K=1,NY
      FU=FY
      KP1=K+1
      KM1=K-1
      IF(KP1.LE.NY) GO TO 120
      KP1=NY
      GO TO 125
  120 IF(KM1.GE.1) GO TO 130
      KM1=1
  125 FU=FU+FU
  130 FP=0.5*FU
      VK=(VH(J,KP1)-VH(J,KM1))*FU
      PK=(PHIH(J,KP1)-PHIH(J,KM1))*FU
      Q(1,K)=1.+VK
      Q(2,K)=PK
      Q(3,K)=0.5*PK
      Q(4,K)=1.+0.5*VK
      R(1,K)=FU*VH(J,K)
      R(4,K)=R(1,K)
      R(2,K)=FP*PHIH(J,K)
      R(3,K)=R(2,K)
      DO 135 L=1,4
  135 P(L,K)=-R(L,K)
      IF(K.EQ.1) GO TO 136
      IF(K.NE.NY) GO TO 140
  136 Q(1,K)=1.
      Q(3,K)=0.
      R(1,K)=0.
      P(1,K)=0.
      P(3,K)=0.
      R(3,K)=0.
  140 CONTINUE
      CALL LUDCOM(P,Q,R,AJ(1,1,J),BJ(1,1,J),CJ(1,1,J),X,Y,NY,NY,0)
  145 CONTINUE
  268 DO 270 K=1,NY
      DO 270 J=1,NX
      V(J,K)=VH(J,K)
  270 PHI(J,K)=PHIH(J,K)
      DO 300 L=1,NINT
      DO 290 J=1,NX
      JP1=J+1
      JM1=J-1
      IF(JP1.GT.NX) JP1=1
      IF(JM1.LT.1) JM1=NX
      DO 280 K=1,NY
      FKT=FF(K)
      FU=FY
      KP1=K+1
      KM1=K-1
      IF(KP1.LE.NY) GO TO 272
      KP1=NY
      GO TO 274
  272 IF(KM1.GE.1) GO TO 275
      KM1=1
  274 FU=FU+FU
  275 FP=0.5*FU
      S(1,K)=VH(J,K)-UH(J,K)*((VH(JP1,K)-VH(JM1,K))*FU+FKT)
      S(2,K)=PHIH(J,K)*(1.-FX2*(UH(JP1,K)-UH(JM1,K)))-FX*UH(J,K)*
     1 (PHIH(JP1,K)-PHIH(JM1,K))
      S(1,K)=V(J,K)*(1.+FU*(V(J,KP1)-V(J,KM1)))-S(1,K)
     1 +FP*PHI(J,K)*(PHI(J,KP1)-PHI(J,KM1))
      S(2,K)=PHI(J,K)*(1.+FP*(V(J,KP1)-V(J,KM1)))+FU*V(J,K)*
     1 (PHI(J,KP1)-PHI(J,KM1))-S(2,K)
      IF(BDJ.EQ.0.) GO TO 280
      S(1,K)=S(1,K)+BDJ* (VH(JP1,K)-2.*VH(J,K)+VH(JM1,K))
      S(2,K)=S(2,K)+BDJ*(PHIH(JP1,K)-2.*PHIH(J,K)+PHIH(JM1,K))
  280 CONTINUE
      S(1,1)=0.
      S(1,NY)=0.
      CALL BAKBLK(AJ(1,1,J),BJ(1,1,J),CJ(1,1,J),P,Q,S,E,NY,NY,0)
      DO 285 K=1,NY
      V(J,K)=V(J,K)-E(1,K)
  285 PHI(J,K)=PHI(J,K)-E(2,K)
      V(J,1)=0.
      V(J,NY)=0.
  290 CONTINUE
  300 CONTINUE
C  OBTAIN U FOR NEXT STEP
      IF(KD.NE.M) GO TO 315
      DO 310 J=1,NX
      DO 305 K=1,NY
      Y(K)=1.
      Z(K)=FY*V (J,K)
  305 X(K)=-Z(K)
      CALL LUTRID(X,Y,Z,AM(1,J),BM(1,J),CM(1,J),P,Q,NY,NY,0)
  310 CONTINUE
  315 DO 316 K=1,NY
      DO 316 J=1,NX
  316 U(J,K)=UH(J,K)
      DO 345 L=1,NINT
      DO 335 J=1,NX
      JP1=J+1
      JM1=J-1
      IF(JP1.GT.NX)JP1=1
      IF(JM1.LT.1) JM1=NX
      DO 325 K=1,NY
      FU=FY
      FKT=FF(K)
      KP1=K+1
      KM1=K-1
      IF(KP1.LE.NY) GO TO 318
      KP1=NY
      GO TO 320
  318 IF(KM1.GE.1) GO TO 322
      KM1=1
  320 FU=FU+FU
  322 FP=0.5*FU
      Z(K)=UH(J,K)*(1.-FX*(UH(JP1,K)-UH(JM1,K)))-FX2*PHIH(J,K)*
     1 (PHIH(JP1,K)-PHIH(JM1,K))
      Z(K)  =U(J,K)+V(J,K)*(FU*(U(J,KP1)-U(J,KM1))-FKT)-Z(K)
       IF(BDJ.EQ.0.) GO TO 325
      Z(K)=Z(K)+BDJ*(UH(JP1,K)-2.*UH(J,K)+UH(JM1,K))
 325  CALL BAKTRI(AM(1,J),BM(1,J),CM(1,J),P,Q,Z,G,NY,NY,0)
      DO 330 K=1,NY
  330 U(J,K)=U(J,K)-G(K)
  335 CONTINUE
  345 CONTINUE
      TIME=TIME+DT
      IGRF=0
  408 CALL LOOK(U,V,PHI,H,NX,NY,LX,F,LY,HT,ZT,ET)
      HDF=ABS(HT-H0)
      ZDF=ABS(ZT-Z0)
      EDF=ABS(ET-E0)
      SECP=SECOND(CP)
      WRITE(NOUT,4015) SECP
 4015 FORMAT(5H TIME,F8.3,7H CP SEC)
C  TEST WHETHER TO ADJUST THE H-FIELD,
C  SO AS TO CORRECT THE TOTAL MASS.
      IF(HDF.LT.EPSH) GO TO 402
      DO 406 K=1,NY
      DO 406 J=1,NX
      H(J,K)=H(J,K)+H0-HT
  406 PHI(J,K)=2.*SQRT(GG*H(J,K))
      GO TO 408
C  TEST WHETHER TO ADJUST ALL THE FIELDS,
C  SO AS TO (AT LEAST PARTIALLY) CORRECT POTENTIAL ENSTROPHY & TOTAL ENERGY
C  - INTEGRAL INVARIANTS CONSERVATION.
  402 IF(ZDF.LT.EPSZ.AND.EDF.LT.EPSE) GO TO 411
      WRITE(NOUT,400) TIME,HT,ZT,ET,HDF,ZDF,EDF
  400 FORMAT(16H0SCALED TIME  = ,F12.2,25H; CONSTRAINTS, H, Z & E: ,3E14
     *.6/23H ERRORS     , H, Z & E : ,3E14.6)
      IGRF=1
C  SET UP AN AUGMENTED LAGRANGIAN WITH MULTIPLIERS UUH, UZ, AND UE, AND
C  PENALTIES PNLTH, PNLTZ, AND PNLTE.
      ETA=100.
      ACC=ETA*SQRT(ZDF**2+EDF**2)
      ACMIN=ACC/10.
      RH=RH0
      RZ=RZ0
      RE=RE0
      UUH=0.
      UZ=0.
      UE=0.
      PNLTH=0.
      PNLTZ=0.5/RZ
      PNLTE=0.5/RE
C  THE FOLLOWING SECTION (UP TO 'GO TO 403') FORMS A LOOP, IN WHICH
C  THE IMSL LIBRARY SUBROUTINE ZXCGR IS CALLED TO MINIMIZE THE AUGMENTED
C  LAGRAGIAN TO AN ACCURACY DEPENDENT ON ETA.  THEREAFTER THE MULTIPLIERS,
C  PENALTIES, AND ETA ARE UPDATED, UNTIL THE CONSTRAINTS ARE SATISFIED TO
C  WITHIN THE REQUIRED ACCURACY, OR UNTIL ETA BECOMES TOO SMALL.
C
C  ZXCGR CALLS FUNCT TO EVALUATE THE FUNCTION VALUE AND GRADIENT VECTOR.
C
C  ALTERNATIVELY THE NAG LIBRARY SUBROUTINE E04DBF, WHICH ALSO USES THE
C  CONJUGATE GRADIENT METHOD TO MINIMIZE THE FUNCTION, AND ALSO REQUIRES
C  FIRST ORDER DERIVATIVES, COULD BE USED.
C  IN THAT CASE THE DIMENSION OF THE VECTOR CONTAINING WORKING SPACE, WS,
C  MAY BE REDUCED TO 540, AN ARRAY XTOL OF LENGTH 540 SHOULD BE DEFINED,
C  AND MONIT SHOULD BE INCLUDED IN THE 'EXTERNAL' STATEMENT.
C  THE CALLING SEQUENCE WOULD BE:
C  CALL E04DBF(NN,XC,FEST,GC,XTOL,WS,FUNCT,MONIT,100,IFAIL)
C  SUBROUTINE MONIT(NN,XC,FEST,GC,NCALL) SHOULD BE INCLUDED.
C  ITS PURPOSE IS MERELY TO PRINT THE PARAMETERS, IF SO DESIRED.
  403 DO 404 II=1,NN
  404 XO(II)=XC(II)
      WRITE(NOUT,405) ETA,RH,RZ,RE,UUH,UZ,UE,PNLTH,PNLTZ,PNLTE
 405 FORMAT(7HOETA = ,F14.6/22H R               , H, Z & E: ,3E14.6/
    *22H U               , H, Z & E: ,3E14.6/22H PENALTIES, H, Z    E: ,3E14.6)
     F0=PNLTZ*ZDF**2+PNLTE*EDF**2
     CALL ZXCGR(FUNCT,NN,ACC,100,F0,XC,GC,FEST,WS,IFAIL)
     WRITE(NOUT,399) IFAIL
 399 FORMAT(8H IFAIL =,I5)
     DO 407 K=1,NY
     DO 407 J=1,NX
 407 PHI(J,K)=2.*SQRT(GG*H(J,K))
     CALL LOOK(U,V,PHI,H,NX,NY,LX,F,LY,HT,ZT,ET)
     HDIF=HT-H0
     ZDIF=ZT-Z0
     EDIF=ET-E0
     WRITE(NOUT,400) TIME,HT,ZT,ET,HDIF,ZDIF,EDIF
     IF(ABS(4.*ZDIF).GT.ZDF) RZ=RZ*0.4
     UZ=UZ+ZDIF/RZ
     PNLTZ=0.5/RZ
     IF(ABS(4.*EDIF).GT.EDF) RE=RE*0.4
     UE=UE+EDIF/RE
     PNLTE=0.5/RE
 410 HDF=ABS(HDIF)
     ZDF=ABS(ZDIF)
     EDF=ABS(EDIF)
     IF(ZDF.LT.EPSZ/10..AND.EDF.LT.EPSE/10.) GO TO 411
     ETA=ETA*0.8
     IF(ETA.LT.40.) GO TO 412
     ACC=AMAX1(ACMIN,ETA*SQRT(ZDF**2+EDF**2))
     ACMIN=ACC/10.
     GO TO 403
 412 WRITE(NOUT,4105)
4105 FORMAT(26H 5 ITERATIONS INSUFFICIENT)
 411 CONTINUE
     IF(IOPT.GE.IPR) GO TO 350
     IF(I.LT.NT) GO TO 355
     WRITE(NOUT,413) TIME
 413 FORMAT(15HOSCALED TIME = ,F12.0)
350   CALL HOUT(H,NX,NY,LX)
     IOPT=0
     CALL MAPPA(H,2.E03,NX,NY,LX)
  355 IF(IGRF.EQ.0) GO TO 356
     IT1=IT1+1
     HC(IT1)=HT/H0
     ZC(IT1)=ZT/Z0
     EC(IT1)=ET/E0
     TC(IT1)=TIME/36.
     IF(HDF.LT.EPSH) GO TO 401
     DO 409 K=1,NY
     DO 409 J=1,NX
     H(J,K)=H(J,K)+H0-HT
  409 PHI(J,K)=2.*SQRT(GG*H(J,K))
     CALL LOOK(U,V,PHI,H,NX,NY,LX,F,LY,HT,ZT,ET)
     HDIF=HT-H0
     ZDIF=ZT-Z0
     EDIF=ET-E0
     WRITE(NOUT,400) TIME,HT,ZT,ET,HDIF,ZDIF,EDIF
  401 CONTINUE
  356 IF(KD.EQ.M) KD=0
  360 KD=KD+1
      WRITE(NTAPE) IT1,(TC(I),I=1,IT1),(HC(I),I=1,IT1),(ZC(I),I=1,IT1),
     *(EC(I),I=1,IT1)
      RETURN
      END
      SUBROUTINE SETUP(U,V,PHI,H,F,NX,NY,S,C,LX,LY)
C  TO SET UP THE INITIAL VALUES OF THE HEIGHT AND VELOCITY FIELDS.
C
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,IND
      COMMON/RITE/NIN,NOUT,NTAPE
      DIMENSION  U(LX,LY),V(LX,LY),PHI(LX,LY),F(LY),S(LX),C(LX),H(LX,LY)
      DATA TUPI/6.2831853071796/
    1 FORMAT(6E10.4)
    3 FORMAT(25H1 SHALLOW WATER EQUATIONS/)
    4 FORMAT(17H0 CONSTANTS:  H0=,F5.0,2H M,10X,5HFHAT=,E9.2,6H/SEC  ,
     1 10X,2HL=,F9.0,2HM=,12X,3HDX=,F8.0,2H M/14X,3HH1=,F5.0,2H M,10X,
     2 5HBETA=,E9.2,6H/SEC/M,10X,2HD=,F9.0,2H M,12X,3HDY=,F8.0,2H M/
     3 14X,3HH2=,F5.0,2H M,40X,2HT=,F9.0,4H SEC,10X,3HDT=,F8.0,4H 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
      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
      Y=Y+DY
   20 FKM=FKM+1.
   24 DO 25 J=1,NX
      V(J,1)=0.
   25 V(J,NY)=0.
      RETURN
      END
      SUBROUTINF MAPPA(FUN,C,NX,NZ,LX)
C  THIS SUBROUTINE PROVIDES A VISUAL DISPLAY OF THE FIELD RY PRINTING AN
C  ISOLINE CONTOUR OF THE FIELD, USING DIGITS FROM 0 TO 9.
C  THE PARAMETER FUN GIVES THE FIELD TO BE CONTOURED, WHILE C IS A
C  PARAMETER GIVING THE INVERSE OF THE CONTOUR CONSTANT.
      DIMENSION FUN(LX,NZ),ANS(4,116),IANS(116),NUM(10)
      COMMON/RITE/NIN,NOUT,NTAPE
      DATA NUM(1)/1H1/,NUM(2)/1H2/,NUM(3)/1H3/,NUM(4)/1H4/,NUM(5)/1H5/
     *,NUM(6)/1H6/,NUM(7)/1H7/,NUM(8)/1H8/,NUM(9)/1H9/,NUM(10)/1H0/,BL/
     *1H /
    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
      LEND=K
      WRITE(NOUT,1) (J,J=1,NZ)
      JB=1
   10 I=I+1
      WRITE(NOUT,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
      JXPN=JX+N
      YDIF=(ANS(L,JXPN)-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)=BLNK
   40 CONTINUE
      IF(L.GT.1) GO TO 45
      WRITE(NOUT,4) (IANS(M),M=1,MEND)
      GO TO 50
   45 WRITE(NOUT,3) (IANS(M),M=1,MEND)
   50 CONTINUE
      IF(I-NX) 10,55,65
   55 LEND=1
      I=I+1
      WRITE(NOUT,2) I
      DO 60 J=1,NZ
      JX=1+N*(J-JB)
 60      ANS(1,JX)=FUN(1,J)
      GO TO 18
   65 WRITE(NOUT,1) (J,J=1,NZ)
      RETURN
      END
      SUBROUTINE UVOUT(W,NX,NY,LX)
C  THIS SUBROUTINE PRINTS OUT THE VALUES OF THE VELOCITY FIELD COMPONENTS
C  IN MATRIX FORM.
C  W STANDS FOR EITHER U OR V COMPONENTS OF THE VELOCITY FIELD.
      DIMENSION W(LX,NY)
      COMMON/RITE/NIN,NOUT,NTAPE
      DATA IND/0/
    1 FORMAT(3H0   ,22I6/)
    2 FORMAT(1X,I2,22F6.2)
      JE=0
 5       JB=JE+1
      JE=MINO(NX ,JE+22)
      WRITE(NOUT,1) (     J,J=JB,JE)
      KK=NY
      DO 10 K=1,NY
      KM=KK-1
      WRITE(NOUT,2) KM,(W(J,KK),J=JB,JE)
 10   KK=KM
      IF(JE.LT.NX) GO TO 5
      RETURN
       END
      SUBROUTINE HOUT(H,NX,NY,LX)
C  THIS SUBROUTINE PRINTS OUT THE HEIGHT FIELD VALUES IN MATRIX FORMAT.
      DIMENSION H(LX,NY)
      COMMON/RITE/NIN,NOUT,NTAPE
      DATA IND/0/
    6 FORMAT(15H0 HEIGHT VALUES/)
    7 FORMAT(3X,15I8/)
    8 FORMAT(1X,I2,15F8.5)
      JE=0
 5    JB=JE+1
      JE=MINO(NX ,JE+15)
      WRITE(NOUT,6)
      WRITE(NOUT,7) (     J,J=JB,JE)
      KK=NY
      DO 10 K=1,NY
      KM=KK-1
      WRITE(NOUT,8)KM,(H(J,KK),J=JB,JE)
 10        KK=KM
      IF(JE.LT.NX) GO TO 5
       RETURN
       FND
      SUBROUTINE LOOK(U,V,PHI,H,NX,NY,LX,F,LY,HMEAN,ZMEAN,ENERGY)
C  THIS SUBROUTINE CALCULATES THE TOTAL ENERGY, TOTAL MASS AND POTENTIAL
C  ENSTROPHY, WHICH ARE INTEGRAL INVARIANTS OF THE SHALLOW WATER EQUATIONS.
C  IT ALSO PRINTS THE VALUES OF THE HEIGHT FIELD BY CALLING SUBROUTINE HOUT.
      COMMON/CONST/FL,D,T,DX,DY,DT,FX,FY,FT, G,TIME,IPR,ADJ,BDJ,IND
      DIMENSION F(LY)
      DIMENSION U(LX,LY),V(LX,LY),PHI(LX,LY),H(LX,LY)
      COMMON/RITE/NIN,NOUT,NTAPE
      DATA JND/0/,NSTEP/0/
      IF(JND.GT.0) GO TO 5
      G4INV=1./(4.*G)
      AREA=NX*(NY-1)
      ECNST=DX*DY/(G+G)
    5 SUMENG=0.
      HMEAN=0.
      FAC=0.5
      DO 40 K=1,NY
      IF(K.EQ.NY) FAC=0.5
      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(JND.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
      HMEAN=HMEAN/AREA
      ENERGY=SUMENG*ECNST
      NY1=NY-1
      E2=DX*DY*0.5
      ZMEAN=0.
      DO 60 K=2,NY1
      ENS=0.
      DO 56 J=1,NX
      JP1=J+1
      JM1=J-1
      IF(J.EQ.1) JM1=NX
      IF(J.EQ.NX) JP1=1
      VX=(V(JP1,K)-V(JM1,K))/(2.*DX)
      UY=(U(J,K+1)-U(J,K-1))/(2.*DY)
      A=VX-UY+F(K)
   56 ENS=ENS+A*A/H(J,K)
   60 ZMEAN=ZMEAN+ENS
      ZMEAN=ZMEAN*E2
      NSTEP=IPR
      IF(JND.NE.0) GO TO 45
      JND=1
      GO TO 50
   45 CONTINUE
   50 RETURN
      END
      SUBROUTINE LUTRID(P,Q,R,A,B,C,D,E,L,N,IND)
C  SIMILAR TO SUBROUTINE LUDCOM (Q.V.) WHICH IS THE 2*2 BLOCK CASE.
C  HERE THE L-U DECOMPOSITION OF A TRIDIAGONAL MATRIX IS PERFORMED.
C     ARRAY P CONTAINS SUB-DIAGONAL LELMENTS.
C     ARRAY Q CONTAINS DIAGONAL ELEMENTS.
C     ARRAY R CONTAINS SUPER-DIAGONAL ELEMENTS.
C     N IS THE ORDER OF THE MATRIX.
C     IF THE MATRIX IS CYCLIC SET IND=1.  OTHERWISE SET IND=0.
C  ON RETURN TO THE CALLING PROGRAM
C     IN THE  L-MATRIX THE DIAGONAL ELEMENTS ARE IN ARRAY A,
C                      THE SUB-DIAGONAL ELEMENTS ARE IN ARRAY B,
C         AND IF IND=1  THE FIRST N-2 ELEMENTS OF ROW N ARE IN ARRAY E.
C     IN THE U-MATRIX THE DIAGONAL ELEMENTS ARE ALL EQUAL TO 1. (NOT STORED),
C                      THE SUPER-DIAGONAL ELEMENTS ARE IN ARRAY C,
C         AND IF IND=1 THE FIRST N-2 ELEMENTS OF COLUMN N ARE IN ARRAY D.
      DIMENSION P(L),Q(L),R(L),A(L),B(L),C(L),D(L),E(L)
      NM=N-1
      DO 60 I=1,N
      IF(I.GT.1) GO TO 20
      A(1)=Q(1)
      GO TO 30
   20 B(I)=P(I)
      IF(I.LT.N) GO TO 28
      IF(IND.EQ.0) GO TO 28
      B(N)=B(N)+E(NM)
   28 IM=I-1
      A(I)=Q(I)-B(I)*C(IM)
      IF(I-NM) 30,29,55
   29 IF(IND.EQ.0) GO TO 30
      C(I)=(R(I)-B(I)*D(IM))/A(I)
      GO TO 40
   30 C(I)=R(I)/A(I)
      IF(IND.EQ.0) GO TO 60
      IF(I.GT.1) GO TO 40
      S1=0.
      E(1)=R(N)
      T1=P(1)
      GO TO 45
   40 E(I)=-E(IM)*C(IM)
      IF(I.EQ.NM) GO TO 60
      T1=-B(I)*D(IM)
   45 D(I)=T1/A(I)
      S1=S1+E(I)*D(I)
      GO TO 60
   55 IF(IND.EQ.0) GO TO 60
      A(N)=A(N)-S1
   60 CONTINUE
      RETURN
      END
      SUBROUTINE BAKBLK(A,B,C,D,E,S,X,L,N,IND)
C  TO FIND  X  WHERE  L*Y=S  AND  U*X=Y
C  IN ORDER TO SOLVE FOR J(INVERSE)*G IN THE QUASI-NEWTON METHOD.
C    L AND U WERE FORMED BY  L-U DECOMPOSITION OF A BLOCK (2*2)
C      TRIDIAGONAL MATRIX OF ORDER  2*N.   IND=1 INDICATES THAT THE
C      ORIGINAL MATRIX IS CYCLIC.
C
C    IN THE  L-MATRIX THE DIAGONAL ELEMENTS ARE IN ARRAY A,
C                     THE SUB-DIAGONAL ELEMENTS ARE IN ARRAY B,
C        AND IF IND=1  THE FIRST N-2 ELEMENTS OF ROW N ARE IN ARRAY E.
```

```
C     IN THE U-MATRIX THE DIAGONAL ELEMENTS ARE ALL EQUAL TO 1. (NOT STORED),
C                         THE SUPER-DIAGONAL ELEMENTS ARE IN ARRAY C,
C           AND IF IND=1 THE FIRST N-2 ELEMENTS OF COLUMN N ARE IN ARRAY D.
      DIMENSION A(4,L),B(4,L),C(4,L),D(4,L),E(4,L),S(2,L),X(2,L)
      NM=N-1
      IF(IND.EQ.0) GO TO 10
      NM2=N-2
      E1=0.
      E2=0.
   10 DO 30 I=1,N
      TEMP=1./(A(1,I)*A(4,I)-A(2,I)*A(3,I))
      IF(I.GT.1) GO TO 15
      T1=S(1,1)
      T2=S(2,1)
      GO TO 20
   15 IM=I-1
      T1=S(1,I)-B(1,I)*X(1,IM)-B(3,I)*X(2,IM)
      T2=S(2,I)-B(2,I)*X(1,IM)-B(4,I)*X(2,IM)
      IF(I.LT.N) GO TO 20
      IF(IND.LE.0) GO TO 20
      T1=T1-E1
      T2=T2-E2
   20 X(1,I)= (A(4,I)*T1-A(3,I)*T2)*TEMP
      X(2,I)=-(A(2,I)*T1-A(1,I)*T2)*TEMP
      IF(I.GE.NM) GO TO 30
      IF(IND.LE.0) GO TO 30
   25 E1=E1+E(1,I)*X(1,I)+E(3,I)*X(2,I)
      E2=E2+E(2,I)*X(1,I)+E(4,I)*X(2,I)
   30 CONTINUE
      IF(IND.LE.0) GO TO 40
      T1=X(1,N)
      T2=X(2,N)
      DO 35 I=1,NM2
      X(1,I)=X(1,I)-D(1,I)*T1-D(3,I)*T2
   35 X(2,I)=X(2,I)-D(2,I)*T1-D(4,I)*T2
   40 K=N
      DO 45 I=2,N
      KP=K
      K=K-1
      X(1,K)=X(1,K)-C(1,K)*X(1,KP)-C(3,K)*X(2,KP)
   45 X(2,K)=X(2,K)-C(2,K)*X(1,KP)-C(4,K)*X(2,KP)
      RETURN
      END
      SUBROUTINE BAKTRI(A,B,C,D,E,S,X,L,N,IND)
C   TO FIND  X  WHERE  L*Y=S  AND  U*X=Y  WHERE
C     L AND U WERE FORMED BY  L-U DECOMPOSITION OF A TRIDIAGONAL MATRIX
C       OF ORDER N.  IND-1 INDICATES THE ORIGINAL MATRIX IS CYCLIC.
C   HERE X STANDS FOR THE VECTOR J(INVERSE)*G IN THE QUASI-NEWTON METHOD.
C     A AND B CONTAIN DIAGONAL AND SUB-DIAGONAL ELEMENTS OF MATRIX L.
C       IF IND=1 THE ELEMENTS I=1 TO I=N-2 OF THE N-TH ROW OF MATRIX L
C         ARE IN ARRAY E
C     C CONTAINS THE SUPER-DIAGONAL ELEMENTS OF MATRIX U.
C       THE DIAGONAL ELEMENTS OF MATRIX U ARE ALL EQUAL TO 1.
C       IF IND=1 THE ELEMENTS I=1 TO I=N-2 OF THE N-TH COLUMN OF MATRIX L
C         ARE IN ARRAY D.
      DIMENSION A(L),B(L),C(L),D(L),E(L),S(L),X(L)
      IEND=N
      IF(IND.GT.0) IEND=IEND-1
C   (1) FIND Y,WHERE  L*Y=S.  (Y IS STORED IN ARRAY X)
C       B(I)*X(I-1)+A(I)*X(I)=S(I)
      X(1)=S(1)/A(1)
      DO 10 I=2,IEND
      IM=I-1
   10 X(I)=(S(I)-B(I)*X(IM))/A(I)
      IF(IND.LE.0) GO TO 30
C     IF IND=1,  SUM(E(I)*X(I),I=1 TO N-2)+B(N)*X(N-1)+A(N)*X(N)=S(N)
      EX=0.
      NM=N-2
      DO 15 I=1,NM
   15 EX=EX+E(I)*X(I)
      X(N)=(S(N)-B(N)*X(N-1)-EX)/A(N)
      XN=X(N)
C   (2) FIND X,WHERE  U*X=Y.
C
C     X(K)+C(K)*X(K+1)+D(K)*X(N)=Y(K)    FOR  K=N-2 TO 1, IF IND=1.
      DO 25 I=1,NM
   25 X(I)=X(I)-D(I)*XN
   30 K=N
      DO 35 I=2,N
      KP=K
      K=K-1
   35 X(K)=X(K)-C(K)*X(KP)
      RETURN
      END
      SUBROUTINE FUNCT(N,XC,FC,GC)
C   THIS SUBROUTINE CALCULATES THE VALUE OF THE AUGMENTED LAGRANGIAN, FC
C   AND ITS DERIVATIVE IN RESPECT OF EACH H(I,J), U(I,J), AND V(I,J),
C   GC(IJK), WHERE IJK TAKES VALUES BETWEEN 1 AND 3*NX*NY.
C   THIS ROUTINE IS USED FOR IMPLEMENTING THE AUGMENTED LAGRANGIAN TECHNIQUE
C   OF NONLINEARLY CONSTRAINED MINIMIZATION TO ENFORCE 'A POSTERIORI'
C   CONSERVATION OF THE INTEGRAL INVARIANTS OF THE SHALLOW WATER EQUATIONS.
      COMMON/OLD/XO(540),HO,ZO,EO,ALPHA,BETA,TG,F(12),NX,NY,LX,LY,PNLTH
     1,PNLTZ,PNLTE,UH,UZ,UE
      COMMON/CONST/FL,D,T,DXS,DYS,DT,FX,FY,FT,G,TIME,IPR,ADJ,BDJ,IND
      DIMENSION XC(540),GC(540),DZ(540),
     1PHIS(15,12),DH(180)
      M12=N/3
      M21=M12+1
      M22=M12+M12
      M31=M22+1
C        CALCULATE FUNCTION VALUE FC
      SUMSQ=0.
      DO 1 II=1,M22
    1 SUMSQ=SUMSQ+ALPHA*(XC(II)-XO(II))**2
      DO 2 II=M31,N
    2 SUMSQ=SUMSQ+BETA*(XC(II)-XO(II))**2
C        CALCULATE E,Z AND H
      IH=M22
      DO 21 J=1,NY
      DO 21 I=1,NX
      IH=IH+1
   21 PHIS(I,J)=2.*SQRT(G*XC(IH))
      CALL LOOK(XC(1),XC(181),PHIS,XC(361),NX,NY,LX,F,LY,HT,ZT,ET)
      HDIF=HT-HO
      EDIF=ET-EO
      ZDIF=ZT-ZO
C
      FC=SUMSQ+PNLTH*HDIF**2+PNLTZ*ZDIF**2+PNLTE*EDIF**2
     *+UH*HDIF+UZ*ZDIF+UE*EDIF
C        CALCULATE DZ/DU AND DZ/DV AND DZ/DH
      TDXS=DXS*2.
      TDYS=DYS*2.
      DYS2=DYS/2.
      DXS2=DXS/2.
      C=(-DXS*DYS)/2.
      NYM1=NY-1
      AREA=1./FLOAT(NX*(NY-1))
      AREA2=AREA/2.
      IU=0
      DO 3  J=1,NY
      DO 3  I=1,NX
      IU=IU+1
      IV=IU+M12
      IH=IU+M22
      DZ(IU)=0.
      DZ(IV)=0.
      DZ(IH)=0.
      DH(IU)=AREA2
      IF(J.EQ.1.OR.J.EQ.NY) GO TO 4
      DH(IU)=AREA
      IHMS=IH-1
      IVMS=IV-1
      IVNS=IVMS-1
      IUMP=IU-1+NX
      IUMM=IU-1-NX
      IUSP=IU+NX
      IUSM=IU-NX
      IF(I.NE.1) GO TO 5
C         CYCLIC X BOUNDS
      IHMS=IHMS+NX
      IVMS=IVMS+NX
      IVNS=IVNS+NX
      IUMP=IUMP+NX
      IUMM=IUMM+NX
    5 IF(I.EQ.2) IVNS=IVNS+NX
      IHPS=IH+1
      IVPS=IV+1
      IVQS=IVPS+1
      IUPP=IU+1+NX
      IUPM=IU+1-NX
      IF(I.NE.NX) GO TO 6
      IHPS=IHPS-NX
      IVPS=IVPS-NX
      IVQS=IVQS-NX
      IUPP=IUPP-NX
      IUPM=IUPM-NX
    6 IF(I.EQ.(NX-1)) IVQS=IVQS-NX
      DZ(IV)=((((XC(IV)-XC(IVNS))/TDXS-(XC(IUMP)-XC(IUMM))/TDYS+F(J))
     1/XC(IHMS))-(((XC(IVQS)-XC(IV))/TDXS-(XC(IUPP)-XC(IUPM))/TDYS+
     2F(J))/XC(IHPS)))*DYS2
      DZ(IH)=C*(((XC(IVPS)-XC(IVMS))/TDXS-(XC(IUSP)-XC(IUSM))/TDYS+
     1F(J))**2)/(XC(IH)**2)
    4 DZUL=0.
      DZUR=0.
      IF(J.GE.NYM1) GO TO 7
      IVPP=IV+1+NX
      IVMP=IV-1+NX
      IF(I.EQ.1) IVMP=IVMP+NX
      IF(I.EQ.NX) IVPP=IVPP-NX
      IUSQ=IU+NX+NX
      IHSP=IH+NX
      DZUL=DXS2*((XC(IVPP)-XC(IVMP))/TDXS-(XC(IUSQ)-XC(IU))/TDYS+
     1F(J+1))/XC(IHSP)
      IF(J.LE.2) GO TO 8
    7 IVPM=IV+1-NX
      IVMM=IV-1-NX
      IF(I.EQ.1) IVMM=IVMM+NX
      IF(I.EQ.NX) IVPM=IVPM-NX
      IUSN=IU-NX-NX
      IHSM=IH-NX
      DZUR=DXS2*((XC(IVPM)-XC(IVMM))/TDXS-(XC(IU)-XC(IUSN))/TDYS+
     1F(J-1))/XC(IHSM)
    8 DZ(IU)=DZUL-DZUR
    3 CONTINUE
C         CALCULATE DF/DU
      TALPHA=2.*ALPHA
      TBETA=2.*BETA
      C2=DXS*DYS
      C3=C2*EDIF
      C1=2.*C3
      C4=UE*DXS*DYS
      C5=C4/2.
      DO 10 IU=1,M12
      IH=IU+M22
   10 GC(IU)=TALPHA*(XC(IU)-XO(IU))+2.*ZDIF*DZ(IU)*PNLTZ+UZ*DZ(IU)
     *+PNLTE*C1*XC(IU)*XC(IH)+C4*XC(IU)*XC(IH)
C        CALCULATE DF/DV
      DO 11 IV=M21,M22
      IH=IV+M12
   11 GC(IV)=TALPHA*(XC(IV)-XO(IV))+2.*ZDIF*DZ(IV)*PNLTZ+UZ*DZ(IV)
     *+PNLTE*C1*XC(IV)*XC(IH)+C4*XC(IV)*XC(IH)
C         CALCULATE DF/DH
      DO 12 IU=1,M12
      IV=IU+M12
      IH=IU+M22
      UVH=XC(IU)**2+XC(IV)**2+TG*XC(IH)
   12 GC(IH)=TBETA*(XC(IH)-XO(IH))+PNLTH*2.*HDIF*DH(IU)+UH*DH(IU)
     1+2.*ZDIF*DZ(IH)*PNLTZ+UZ*DZ(IH)+PNLTE*UVH*C3+C5*UVH
      RETURN
      END
C     SUBROUTINE MONIT(N,XC,FC,GC,NCALL)
C     COMMON/OLD/XO(540),H0,Z0,E0,ALPHA,BETA,TG,F(12),NX,NY,LX,LY,PNLTH
C    *,PNLTZ,PNLTE,UH,UZ,UE
C     COMMON/RITE/NIN,NOUT,NTAPE
C     DIMENSION XC(540),GC(540)
C     SUMSQ=0.
C     NXNY=NX*NY
C     GNORM=0.
C     DO 1 IU=1,NXNY
C     IV=IU+NXNY
C     IH=IV+NXNY
C     SUMSQ=SUMSQ+ALPHA*((XC(IU)-XO(IU))**2+(XC(IV)-XO(IV))**2)+BETA*
C    1(XC(IH)-XO(IH))**2
C     GNORM=GNORM+GC(IU)**2+GC(IV)**2+GC(IH)**2
C   1 CONTINUE
C     GNORM=SQRT(GNORM)
C     WRITE(NOUT,2) NCALL,SUMSQ,GNORM
C   2 FORMAT(15H0SUMSQ AT CALL ,I5,3H = ,E12.4,11H , GNORM = ,E12.4)
C     RETURN
C     END
 11.45.44.UCLP, 50, 042,       2.772KLNS.
```