Computers & Geosciences Vol. 12, No. 2, pp. 129–150, 1986
Printed in Great Britain.

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

SHALL4—AN IMPLICIT COMPACT FOURTH-ORDER FORTRAN PROGRAM FOR SOLVING THE SHALLOW-WATER EQUATIONS IN CONSERVATION-LAW FORM

I. M. NAVON[†] and H. A. RIPHAGEN
National Research Institute for Mathematical Sciences, CSIR, P.O. Box 395, Pretoria 0001, South Africa

(*Received* 30 *November* 1984; *revised* 13 *September* 1985)

Abstract—A FORTRAN IV computer program is documented implementing a compact fourth-order accurate finite difference scheme in a spatially factored form, for solving the nonlinear shallow-water equations on a limited domain. In contrast to the usual fourth-order schemes this compact fourth-order scheme requires the solution of only either block-tridiagonal or cyclic block-tridiagonal coefficient matrices. Moreover this compact fourth-order scheme is related to the finite-element method and has a smaller truncation error than the usual fourth-order schemes. The integral invariants of the shallow-water equations are calculated at each time-step and were determined to be well conserved during the numerical integration, ensuring that a realistic nonlinear structure is obtained.

A Schumann–Wallington low-pass filtering procedure was incorporated in the program to overcome the increased aliasing due to the higher accuracy method. A third-order boundary condition is imposed, preserving the overall fourth-order convergence rate of the interior approximation.

Key Words: Shallow-water equations, compact differencing, conservation-law form, fourth-order implicit-finite differences.

INTRODUCTION

This paper describes the computer implementation of the implicit compact fourth-order method for solving the nonlinear shallow-water equations in conservation-law form, and was described in detail by Navon and Riphagen (1979).

This method requires only three-grid-point finite difference expressions instead of the five-grid-point expressions required by the usual fourth-order methods. Moreover it has a truncation error which is smaller by a factor of six than that of the standard five-grid-point fourth-order approximations. The new method also has features common to both finite-difference and finite-element methods (see Cullen, 1975; Cullen, 1977; Cullen and Morton, 1980), and offers a computationally efficient finite-difference alternative to the finite-element approach, because the linear systems to be solved are tridiagonal, in contrast to the more complex coefficient matrices generated by the finite-element method.

Morton (1977) points out that the fourth-order compact method approximates "half-lumping" the mass-matrix of the finite-element method for regular linear elements in the single-stage Galerkin approach. In this program the method is used for solving the nonlinear shallow-water equations which may serve as a test-study in atmospheric and oceanic applications.

In the first section of this paper the system of nonlinear shallow-water equations in conservation-law form is described; as is the test problem used.

In the second section descriptions are given of the fourth-order compact alternating-direction implicit algorithm itself, of the implementation of boundary conditions, and of the low-pass filtering technique used to prevent nonlinear aliasing effects. For further details the reader is referred to Navon and Riphagen (1979).

Finally, the third and last section contains a detailed description of the program SHALL4, its input and output specifications and the different user options.

THE SHALLOW-WATER EQUATIONS IN CONSERVATION-LAW FORM

We consider the shallow-water equations, that is the primitive equations for an incompressible, inviscid fluid with a free surface confined to a channel corresponding to a middle-latitude band. The north and south boundaries are rigid walls, whereas the flow is assumed to be periodic in the east-west direction.

The beta plane approximation is made.

The basic nonlinear shallow-water equations in Eulerian form are

$$\left.\begin{aligned}
&\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} - fv + g\frac{\partial h}{\partial x} = 0\\
&\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + fu + g\frac{\partial h}{\partial y} = 0\\
&\frac{\partial h}{\partial t} + u\frac{\partial h}{\partial x} + v\frac{\partial h}{\partial y} + h\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) = 0
\end{aligned}\right\} \quad (1)$$

for a rectangular domain, $0 \leq x \leq L$, $0 \leq y \leq D$, $t \geq 0$. Variables are defined as follows:

x, y east–west and north–south coordinates, respectively;

t time;

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

uv velocity components in the x and y directions, respectively [$u = u(x,y,t)$, $v = v(x,y,t)$];
h depth of the fluid;
g acceleration of gravity, constant;
f Coriolis force [$= \hat{f} + \beta(y - D/2), \hat{f}, \beta$ constant].

Following Houghton, Kasahara, and Washington (1966), one can write (1) in conservation-law form (i.e. divergence form) as

$$\left.\begin{aligned} &\frac{\partial}{\partial t}(hu) + \frac{\partial}{\partial x}(hu^2) + \frac{\partial}{\partial y}(huv) + gh\frac{\partial h}{\partial x} - fvh = 0 \\ &\frac{\partial}{\partial t}(hv) + \frac{\partial}{\partial x}(huv) + \frac{\partial}{\partial y}(hv^2) + gh\frac{\partial h}{\partial y} + fuh = 0 \\ &\frac{\partial h}{\partial t} + \frac{\partial}{\partial x}(hu) + \frac{\partial}{\partial y}(hv) = 0 \end{aligned}\right\} \tag{2}$$

or in matrix form as

$$\frac{\partial U}{\partial t} + \frac{\partial P}{\partial x} + \frac{\partial Q}{\partial y} - fR = 0, \tag{3}$$

where U, P, Q, and R are the column matrices:

$$U = \begin{bmatrix} \tilde{m} \\ \tilde{n} \\ h \end{bmatrix}, \quad P = \begin{bmatrix} \frac{\tilde{m}^2}{h} + \frac{1}{2}gh^2 \\ \frac{\tilde{m}\tilde{n}}{h} \\ \tilde{m} \end{bmatrix},$$

$$Q = \begin{bmatrix} \frac{\tilde{m}\tilde{n}}{h} \\ \frac{\tilde{n}^2}{h} + \frac{1}{2}gh^2 \\ \tilde{n} \end{bmatrix}, \quad R = \begin{bmatrix} \tilde{n} \\ \tilde{m} \\ 0 \end{bmatrix} \tag{4}$$

in which

$$m = hu \qquad n = hv \tag{5}$$

Periodic boundary conditions are assumed in the x-direction, whereas in the y-direction the boundary condition is

$$v(x,0,t) = v(x,D,t) = 0 \tag{6}$$

and the initial condition

$$U(x,y,0) = \psi(x,y). \tag{7}$$

With these initial and boundary conditions the total energy

$$E = \frac{1}{2}\int_0^L\int_0^D (u^2 + v^2 + gh)h \, dx \, dy \tag{8}$$

is independent of time.

Also independent of time are the average values of the height of the free surface (which is proportional to the total mass):

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

and the enstrophy

$$Z = \iint \left(\frac{q^2}{h}\right) dx \, dy \tag{10}$$

where

$$q = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} + f. \tag{11}$$

For the test problem we decided to use two different initial conditions employed by Grammultvedt (1969), both describing a westerly jet flow with north–south perturbations of different wavelengths and amplitudes along the zonal axis of the jet. These initial conditions have been employed by a considerable number of research workers and thus provide a basis for comparison. The initial height fields are

$$\text{(I)}\ h(x,y) = H_0 + H_1 \tanh\left(\frac{9(D/2 - y)}{2D}\right) + H_2 \operatorname{sech}^2\left(\frac{9(D/2 - y)}{D}\right) \sin\frac{2\pi x}{L} \tag{12}$$

$$\text{(II)}\ h(x,y) = H_0 + H_1 \tanh\left(\frac{9(D/2 - y)}{2D}\right) + H_2 \operatorname{sech}^2\left(\frac{9(D/2 - y)}{D}\right) \cdot \left[0.7 \sin\left(\frac{2\pi x}{L}\right) + 0.6 \sin\frac{6\pi x}{L}\right]. \tag{13}$$

Initial condition I initially has energy only in wave number 1 in the x-direction, whereas initial condition II initially contains energy in wavenumbers 1 and 3 in the x-direction.

The dimensions of the rectangular domain were $L = 4400$ km and $D = 6000$ km, and the following constants were adopted:

$$\begin{aligned} &H_0 = 2000 \text{ m}; \quad H_1 = 220 \text{ m}; \quad H_2 = 133 \text{ m}; \\ &g = 10 \text{ m s}^{-2}; \quad \hat{f} = 10^{-4} \text{ s}^{-1}; \\ &\beta = 1.5\ 10^{-11} \text{ m}^{-1} \text{ s}^{-1}; \end{aligned} \tag{14}$$

where

$$f = \hat{f} + \beta(y - D/2). \tag{15}$$

The time and space increments used were

$$\Delta x = \Delta y = 200 \text{ km}; \quad \Delta t = 900 \text{ s};$$
$$\Delta x = \Delta y = 500 \text{ km}; \quad \Delta t = 1800 \text{ s}. \tag{16}$$

THE BASIC ALGORITHM

Time-differencing and linearization

Denoting by a superscript n the time level $n\ \Delta t$, where Δt is the time increment, we start by using a trapezoidal time-differencing scheme (Beam and Warming, 1976; Briley and McDonald, 1977):

$$U^{n+1} = U^n + \frac{\Delta t}{2}\left[\left(\frac{\partial U}{\partial t}\right)^n + \left(\frac{\partial U}{\partial t}\right)^{n+1}\right] + 0(\Delta t^3). \tag{17}$$

If the scheme (17) is applied to (3), one obtains

$$U^{n+1} = U^n - \frac{\Delta t}{2}\left[\left(\frac{\partial P}{\partial x} + \frac{\partial Q}{\partial y} - fR\right)^n + \left(\frac{\partial P}{\partial x} + \frac{\partial Q}{\partial y} - fR\right)^{n+1}\right] + 0(\Delta t^3). \tag{18}$$

As

$$P^{n+1} = P(U^{n+1}) \quad \text{and} \quad Q^{n+1} = Q(U^{n+1}) \tag{19}$$

are nonlinear functions of U^{n+1}, a linearization procedure (see Steger, 1978; Warming and Beam, 1978) involving a local Taylor expansion about U^n, is employed to overcome the nonlinearity of the problem:

$$\left.\begin{aligned} P^{n+1} &= P^n + A^n(U^{n+1} - U^n) + 0(\Delta t^2) \\ Q^{n+1} &= Q^n + B^n(U^{n+1} - U^n) + 0(\Delta t^2) \end{aligned}\right\}, \tag{20}$$

where the matrices

$$A = \frac{\partial P}{\partial U}, \quad B = \frac{\partial Q}{\partial U} \tag{21}$$

are Jacobian matrices with elements

$$\left(\frac{\partial P}{\partial U}\right)_{qr} = \frac{\partial P_q}{\partial U_r} \quad \text{and} \quad \left(\frac{\partial Q}{\partial U}\right)_{qr} = \frac{\partial Q_q}{\partial U_r}. \tag{22}$$

Substituting (20) into (18), a linear system for U^{n+1} is obtained:

$$\begin{aligned} &\left\{I + \frac{\Delta t}{2}\left[\frac{\partial}{\partial x}(A^n \cdot) + \frac{\partial}{\partial y}(B^n \cdot)\right]\right\}U^{n+1} - \frac{\Delta t}{2}fR^{n+1} \\ &= \left\{I + \frac{\Delta t}{2}\left[\frac{\partial}{\partial x}(A^n \cdot) + \frac{\partial}{\partial y}(B^n \cdot)\right]\right\} \\ &\cdot U^n - \Delta t\left(\frac{\partial P}{\partial x} + \frac{\partial Q}{\partial y}\right)^n + \frac{\Delta t}{2}fR^n. \end{aligned} \tag{23}$$

In Equation (23) and throughout the paper the notation

$$\left[\frac{\partial}{\partial x}(A^n \cdot) + \frac{\partial}{\partial y}(B^n \cdot)\right]U^{n+1} \tag{24}$$

is used to denote

$$\frac{\partial}{\partial x}(A^n U^{n+1}) + \frac{\partial}{\partial y}(B^n U^{n+1}), \tag{25}$$

and I is the unit matrix.

The alternating direction implicit factorization

As it stands, Equation (23) seems to suggest that a large number of operations are required to solve the implicit equations. Clearly, if one could factor the space-difference operators into separate spatial variables, instead of having to solve a formidable matrix inversion problem, one would have only to solve block-tridiagonal systems, using efficient solution algorithms. This significant improvement in efficiency for multidimensional implicit methods is achieved by using the alternating-direction implicit (ADI) algorithm (see Douglas and Gunn, 1964). We first note that in (23) the term fR can be written

$$fR = f\begin{bmatrix} \tilde{n} \\ -\tilde{m} \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & f & 0 \\ -f & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} \tilde{m} \\ \tilde{n} \\ h \end{bmatrix} = CU. \tag{26}$$

Therefore one can write (23) as

$$\begin{aligned} &\left\{I + \frac{\Delta t}{2}\left[\frac{\partial}{\partial x}(A^n \cdot) + \frac{\partial}{\partial y}(B^n \cdot) - C\right]\right\}U^{n+1} \\ &= \left\{I + \frac{\Delta t}{2}\left[\frac{\partial}{\partial x}(A^n \cdot) + \frac{\partial}{\partial y}(B^n \cdot) + C\right]\right\}U^n \\ &\quad - \Delta t\left(\frac{\partial P}{\partial x} + \frac{\partial Q}{\partial y}\right) + 0(\Delta t^3). \end{aligned} \tag{27}$$

The form (27) suggests that we establish a factorizable term within the braces by adding the following third-order perturbation terms:

$$\left.\begin{aligned} &\text{(I)} \quad \frac{\Delta t^3}{4}\frac{\partial}{\partial x}(A^n \cdot)\frac{\partial}{\partial y}(B^n \cdot)\frac{(U^{n+1} - U^n)}{\Delta t} \\ &\qquad = \frac{\Delta t^3}{4}\frac{\partial}{\partial x}(A^n \cdot)\frac{\partial}{\partial y}(B^n \cdot)\frac{\partial}{\partial t}U^n + 0(\Delta t^4) \\ &\text{(II)} \quad \frac{\Delta t^3}{4}C^{(1)}C^{(2)}\frac{(U^{n+1} - U^n)}{\Delta t} \\ &\qquad = \frac{\Delta t^3}{4}C^{(1)}C^{(2)}\frac{\partial}{\partial t}U^n + 0(\Delta t^4) \\ &\text{(III)} \quad \frac{\Delta t^3}{4}\frac{\partial}{\partial x}(A^n \cdot)C^{(2)}\frac{(U^{n+1} + U^n)}{\Delta t} \\ &\text{(IV)} \quad \frac{\Delta t^3}{4}\frac{\partial}{\partial y}(B^n \cdot)C^{(1)}\frac{(U^{n+1} + U^n)}{\Delta t} \end{aligned}\right\} \tag{28}$$

where

$$C^{(1)} = \begin{bmatrix} 0 & 0 & 0 \\ -f & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad C^{(2)} = \begin{bmatrix} 0 & f & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{29}$$

$$C^{(1)} + C^{(2)} = C\,. \tag{30}$$

A scale analysis shows that the last three perturbation terms (II)–(IV) are the order 10^{-8}, 10^{-4} and 10^{-4}, respectively, compared with the magnitude of the first perturbation term. (Typical magnitudes are $h = 2000$ m, $u = 30$ m s^{-1}, $v = 5$ m s^{-1} and $f = 10^{-4}$ s^{-1}.)

The factored scheme then can be written as

$$\begin{aligned} &\left\{I + \frac{\Delta t}{2}\left[\frac{\partial}{\partial x}(A^n\,\cdot) - C^{(1)}\right]\right\} \\ &\quad\left\{I + \frac{\Delta t}{2}\left[\frac{\partial}{\partial y}(B^n\,\cdot) - C^{(2)}\right]\right\}U^{n+1} \\ &\quad = \left\{I + \frac{\Delta t}{2}\left[\frac{\partial}{\partial x}(A^n\,\cdot) + C^{(1)}\right]\right\} \\ &\quad\left\{I + \frac{\Delta t}{2}\left[\frac{\partial}{\partial y}(B^n\,\cdot) + C^{(2)}\right]\right\}U^n \\ &\qquad - \Delta t\left(\frac{\partial P}{\partial x} + \frac{\partial Q}{\partial y}\right)^n . \end{aligned} \tag{31}$$

The Jacobian matrices A and B are given by

$$A = \begin{bmatrix} 2u & 0 & -u^2 + gh \\ v & u & -uv \\ 1 & 0 & 0 \end{bmatrix}, \tag{32}$$

$$B = \begin{bmatrix} v & u & -uv \\ 0 & 2v & -v + gh \\ 0 & 1 & 0 \end{bmatrix}, \tag{33}$$

$$BU = \begin{bmatrix} huv \\ hv^2 + gh^2 \\ hv \end{bmatrix} = \begin{bmatrix} \dfrac{\bar{n}\bar{m}}{h} \\ \dfrac{\bar{n}^2}{h} + gh^2 \\ \bar{n} \end{bmatrix} = Q + \begin{bmatrix} 0 \\ \frac{1}{2}gh^2 \\ 0 \end{bmatrix}, \tag{34}$$

$$AU = \begin{bmatrix} hu^2 + gh^2 \\ huv \\ hu \end{bmatrix} = \begin{bmatrix} \bar{m}^2 + gh^2 \\ \dfrac{\bar{m}\bar{n}}{h} \\ \bar{m} \end{bmatrix} = P + \begin{bmatrix} \frac{1}{2}gh^2 \\ 0 \\ 0 \end{bmatrix}. \tag{35}$$

A computationally convenient form of (31), which emphasizes the spatial splitting, is

$$\bar{U}^{n+1} = \left\{I + \frac{\Delta t}{2}\left[\frac{\partial}{\partial y}(B^n\,\cdot) + C^{(2)}\right]\right\}U^n, \tag{36a}$$

$$\begin{aligned} &\left\{I + \frac{\Delta t}{2}\left[\frac{\partial}{\partial x}(A^n\,\cdot) - C^{(1)}\right]\right\}\bar{\bar{U}}^{n+1} \\ &\quad = \left\{I + \frac{\Delta t}{2}\left[\frac{\partial}{\partial x}(A^n\,\cdot) + C^{(1)}\right]\right\}\bar{U}^{n+1} \\ &\qquad - \Delta t\left(\frac{\partial P}{\partial x} + \frac{\partial Q}{\partial y}\right)^n, \end{aligned} \tag{36b}$$

$$\left\{I + \frac{\Delta t}{2}\left[\frac{\partial}{\partial y}(B^n\,\cdot) - C^{(2)}\right]\right\}U^{n+1} = \bar{\bar{U}}^{n+1} \tag{36c}$$

The introduction of compact fourth-order spatial differencing

For the approximation of the first spatial derivative, the fourth-order compact spatial differencing takes the form

$$\left(\frac{\partial u}{\partial x}\right)_i = \left[\frac{D_{0x}}{(1 + \Delta x^2 D + xD - x/6}\right]u_i + 0(\Delta x^4), \tag{37}$$

and involves only the grid points $i + 1, i, i - 1$ ($x_i = i\,\Delta x$), where

$$\left.\begin{aligned} D_{0x}u_i &= (u_{i+1} - u_{i-1})/2\,\Delta x \\ D_{+x}u_i &= (u_{i+1} - u_i)/\Delta x \\ D_{-x}u_i &= (u_i - u_{i-1})/\Delta x \end{aligned}\right\}. \tag{38}$$

Equation (37) is equivalent to

$$\frac{1}{6}\left[\left(\frac{\partial u}{\partial x}\right)_{i+1} + 4\left(\frac{\partial u}{\partial x}\right)_i + \left(\frac{\partial u}{\partial x}\right)_{i-1}\right] = D_{0x}u_i\,. \tag{39}$$

Thus $(\partial u/\partial x)_i, i = 1, \ldots, N_x$, can be determined from u_i by solving a system of linear equations whose coefficient matrix is tridiagonal and of the form

$$J = \frac{1}{6}\begin{bmatrix} 4 & 1 & & & \\ 1 & 4 & 1 & & 0 \\ & \ddots & \ddots & \ddots & \\ 0 & & \ddots & \ddots & 1 \\ & & & 1 & 4 \end{bmatrix}_{(N_x \times N_x)} \tag{40}$$

For Equation (37), Climent and Leventhal (1975) introduced the more convenient notation

$$\left(\frac{\partial u}{\partial x}\right)_i = \left[\frac{D_{0x}}{(1 + \delta_x^2/6)}\right]u_i = Q_x^{-1}D_{0x}u_i + 0(\Delta x^4) \tag{41}$$

$$\left.\begin{aligned} Q_x u_i &= (1 + \delta_x^2/6)u_i = \frac{1}{6}(u_{i+1} + 4u_i + u_{i-1}) \\ \delta_x^2 u_i &= u_{i+1} - 2u_i + u_{i-1} \end{aligned}\right\} \tag{42}$$

Application of compact fourth-order differencing to the first space derivative in the ADI shallow-water algorithm (36a)–(36c) yields

$$\overline{U}_{ij}^{n+1} = \left\{ I + \frac{\Delta t}{2}[Q_y^{-1}D_{oy}(B_{ij}^n \cdot) + C_{ij}^{(2)}] \right\} U_{ij}^n , \quad (43a)$$

$$\left\{ I + \frac{\Delta t}{2}[Q_x^{-1}D_{ox}(A_{ij}^n \cdot) - C_{ij}^{(1)}] \right\} \overline{\overline{U}}_{ij}^{n+1}$$
$$= \left\{ I + \frac{\Delta t}{2}[Q_x^{-1}D_{ox}(A_{ij}^n \cdot) + C_{ij}^{(1)}] \right\} \overline{U}_{ij}^{n+1}$$
$$- \Delta t(Q_x^{-1}D_{ox}P_{ij}^n + Q_y^{-1}D_{oy}Q_{ij}^n) , \quad (43b)$$

$$\left\{ I + \frac{\Delta t}{2}[Q_y^{-1}D_{oy}(B_{ij}^n \cdot) - C_{ij}^{(2)}] \right\} U_{ij}^{n+1}$$
$$= \overline{\overline{U}}_{ij}^{n+1}, \quad (43c)$$
$$i = 1, \ldots, N_x ,$$
$$j = 1, \ldots, N_y ,$$

where $N_x \Delta_x = L, N_y \Delta_y = D$.

Computational procedure

To evaluate (43a) we write it in the form

$$\overline{U}^{n+1} = \left[I + \frac{\Delta t}{2} C^{(2)} \right] U^n + \frac{\Delta t}{2} Q_y^{-1} D_{oy}(B^n U^n) . \quad (44)$$

For this one-dimensional problem we first solve a block-tridiagonal system

$$Q_y W^n = D_{oy}(B^n U^n) \quad (45)$$

to obtain

$$W^n = Q_y^{-1} D_{oy}(B^n U^n) . \quad (46)$$

In the block-tridiagonal system the individual blocks are (3 × 3). We then evaluate

$$\overline{U}^{n+1} = \left[I + \frac{\Delta t}{2} C^{(2)} \right] U^n + \frac{\Delta t}{2} W^n \quad (47)$$

and with the definition

$$a_2 = \Delta t/2 , \quad (48)$$

we obtain

$$\begin{bmatrix} \overline{U}_1^{n+1} \\ \overline{U}_2^{n+1} \\ \overline{U}_3^{n+1} \end{bmatrix}_{ij} = \begin{bmatrix} h^n u^n + a_2 f h^n v^n + a_2 W_1^n \\ h^n v^n + \qquad a_2 W_2^n \\ h^n \quad + \qquad a_2 W_3^n \end{bmatrix}_{ij} . \quad (49)$$

For Equation (43b) we start by evaluating the right-hand term

$$-\Delta t \frac{\partial Q^n}{\partial y} = -\Delta t Q_y^{-1} D_{oy} Q^n . \quad (50)$$

We define

$$Y = Q_y^{-1} D_{oy} Q^n \quad (51)$$

and solve the block-tridiagonal system

$$Q_y Y = D_{oy} Q^n . \quad (52)$$

We can then write the right-hand side of (43b) as

$$\overline{V}^{n+1} = \left[I + \frac{\Delta t}{2} C^{(1)} \right] \overline{\overline{U}}^{n+1} - \Delta t Y + \frac{\Delta t}{2}(Q_x^{-1} D_{ox}(A^n \overline{U}^{n+1} - 2P^n)) . \quad (53)$$

Multiplying (43b) from the left by the operator Q_x, we then obtain

$$\left[Q_x + \frac{\Delta t}{2}(D_{ox}(A^n \cdot) - Q_x C^{(1)} \right] \overline{U}^{n+1} = Q_x \overline{V}^{n+1}$$
$$= \left[Q_x + \frac{\Delta t}{2} Q_x C^{(1)} \right] \overline{U}^{n+1} - \frac{\Delta t}{2} Q_x Y + \frac{\Delta t}{2}[D_{ox}(A^n \overline{U}^{n+1} - 2P^n)] . \quad (54)$$

Here, owing to the cyclic boundary conditions in the x-direction, cyclic block-tridiagonal systems have to be solved for each $j = 1, \ldots, N_y$.

Efficient algorithms for solving cyclic tridiagonal systems were proposed by Temperton (1975), Navon (1977), and Hindmarsh (1977), among others, and were generalized to block-cyclic tridiagonal matrices by Navon (1977).

For a given j, the cyclic block-tridiagonal matrix resulting from the discretization of Equation (54) has the form

$$R = \begin{bmatrix} E_1 & F_1 & & 0 & D_1 \\ D_2 & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & 0 & \ddots & \ddots & F_{N_x-1} \\ F_{N_x} & & & D_{N_x} & E_{N_x} \end{bmatrix} \quad (55)$$

with

$$D_{ij} = \begin{bmatrix} 1 - 2\alpha u & 0 & -\alpha(-u^2 + gh) \\ -\alpha v + \dfrac{\Delta t f}{2} & 1 - \alpha u & \alpha u v \\ -\alpha & 0 & 1 \end{bmatrix}_{ij}^{(n+1)} , \quad (56)$$

$$E_{ij} = \begin{bmatrix} 4 & 0 & 0 \\ 2\Delta t f & 4 & 0 \\ 0 & 0 & 4 \end{bmatrix}_{ij}^{(n+1)} , \quad (57)$$

$$F_{ij} = \begin{bmatrix} 1 + 2\alpha u & 0 & \alpha(-u^2 + gh) \\ \alpha v + \dfrac{\Delta t f}{2} & 1 + \alpha u & -\alpha u v \\ \alpha & 0 & 1 \end{bmatrix}_{ij}^{(n+1)} , \quad (58)$$

where $\alpha = 6\Delta t/4\Delta x$.

Having obtained $\overline{\overline{U}}^{n+1}$, we finally multiply (43c) from the left by the operator Q_y to obtain

$$Q_y U^{n+1} + \frac{\Delta t}{2} D_{oy}(B^n U^{n+1}) - \frac{\Delta t}{2} Q_y(C^{(2)} U^{n+1}) = Q_y \overline{\overline{U}}^{n+1}. \quad (59)$$

A block-tridiagonal matrix with (3 × 3) individual blocks of dimension N_y has to be inverted for each $i = 1, \ldots, N_x$ at each time-step.

For given i and j the (3 × 3) blocks have the following entries

$$D_{ij} = \begin{bmatrix} 1 - \alpha v - \alpha u - \frac{\Delta t f}{2} & \alpha u v \\ 0 & 1 - 2\alpha v & -\alpha(-v^2 + gh) \\ 0 & -\alpha & 1 \end{bmatrix}_{ij}, \quad (60)$$

$$E_{ij} = \begin{bmatrix} 4 & -2\Delta t f & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 4 \end{bmatrix}_{ij}, \quad (61)$$

$$F_{ij} = \begin{bmatrix} 1 + \alpha v & \alpha u - \Delta t f & -\alpha u v \\ 0 & 1 + 2\alpha v & \alpha(-v^2 + gh) \\ 0 & \alpha & 1 \end{bmatrix}_{ij}. \quad (62)$$

The inverse of a (3 × 3) matrix was calculated explicitly to increase the efficiency of the program.

Implementation of the boundary conditions

In this work we implemented the Adam (1977) $O(h^3)$ boundary conditions and use a convergence result due to Gustafsson (1975) according to which, provided the scheme is stable with regard to boundary conditions, it is possible to use at the boundaries approximations one order lower in accuracy and yet retain the order of convergence of the more accurate interior approximation.

In the y direction we first used the Adam (1977) boundary conditions. For instance, for

$$\frac{\partial}{\partial y}(B^n U^n)_{ij} = Q_y^{-1} D_{oy}(B^n U^n)_{ij} = \overline{W}_{ij}^{n+1}, \quad (63)$$

we wrote

$$Q_y \overline{W}_{ij}^{n+1} = D_{oy}(B^n U^n)_{ij} \quad (64)$$

and, at $j = 1$,

$$\overline{W}_{i,1}^{n+1} + 2\overline{W}_{i,2}^{n+1} = \frac{1}{2\Delta y}(-5(B^n U^n)_{i,1} + 4(B^n U^n)_{i,2} + (B^n U^n)_{i,3}) + O(h^3),$$

$$2\overline{W}_{i,2}^{n+1} + \overline{W}_{i,3}^{n+1} = \frac{1}{2\Delta y}((B^n U^n)_{i,1} - 4(B^n U^n)_{i,2} + 5(B^n U^n)_{i,3}) + O(h^3), \quad (65)$$

and the analog at $j = N_y$, i.e.

$$\overline{W}_{i,N_y}^{n+1} + 2\overline{W}_{i,N_y-1}^{n+1} = \frac{1}{2\Delta y}(5(B^n U^n)_{i,N_y} - 4(B^n U^n)_{i,N_y-1} - (B^n U^n)_{i,N_y-2}) + O(h^3),$$

$$\overline{W}_{i,N_y-2}^{n+1} + \overline{W}_{i,N_y-1}^{n} = \frac{1}{2\Delta y}(-5(B^n U^n)_{i,N_y-2} + 4(B^n U^n)_{i,N_y-1} + (B^n U^n)_{i,N_y}) + O(h^3). \quad (66)$$

For (43c), however, both the value of the derivative and that of the unknown U^{n+1} are required at the y boundaries. Here, we used an inward–backward extrapolation formula for the unknown U_b^{n+1}, due to Gustafsson, Kreiss, and Sundstrom (1972) which has been shown to be stable by Elvius and Sundstrom (1973)

$$(U)_b^{n+1} = 2(U)_{b-1}^n - U_{b-2}^{n-1}, \quad (67)$$

where b denotes the boundary grid point.

Prevention of nonlinear aliasing effects

Owing to the larger aliasing error introduced by the fourth-order accurate scheme a Shuman (1957) filter is applied successively in the x and y directions. It consists of the periodic successive application of the following two-point operators:

$$\overline{U}_i = 3.8798 U_i - 1.77097(U_{i+1} + U_{i-1}) + 0.331065(U_{i+2} + U_{i-2}),$$

$$\overline{U}_i = 0.375\overline{U}_i + 0.25(\overline{U}_{i+1} + \overline{U}_{i231}) + 0.0625(\overline{U}_{i+2} + \overline{U}_{i-2}). \quad (68)$$

This filter completely eliminates waves with wavelengths less than $3\Delta x$, which are the waves contributing to the aliasing effect. A detailed account of the filtering effect is given in Navon and Riphagen (1979).

Comparative results

The compact fourth-order method, used by Navon and Riphagen (1979) has been tested extensively and used by many researchers, among them Cohn and others (1985), Chang and Shirer (1985), Bates (1984), and Takano and Wurtele (1982). It also has been included in a book by Haltiner and Williams (1980). The advantage of the method is the increased accuracy compared to the classical 5-grid-point fourth-order method (the leading truncation error term is $\Delta X^4/180$ instead of $\Delta x^4/30$ for the usual fourth-order methods).

Another advantage is that fewer fictitious boundary points are needed (only 3) than with usual fourth-order methods which require 5 points.

For explicit time-differencing the compact scheme is more expensive as it requires the solution of a tridiagonal system but for implicit time schemes which require system solution anyway—the added computation is of little consequence.

When applied to the advection equation the compact fourth-order scheme is equivalent exactly to the finite-element method with piecewise linear basis function which is known to be considerably more accurate than the 5-point fourth-order scheme (see also Navon 1979b; Navon, 1983).

As far as conservation of integral invariants is concerned Takano and Wurtele (1982) presented a fourth-order energy and potential enstrophy difference scheme which preserved better potential enstrophy and total energy at the cost of lengthy calculations.

A comparison carried out by Navon and Riphagen (1979) on a realistic initial condition — showed the results of the compact fourth-order method to match results of a finite-element integration and to give better results than similar integrations using conventional 5-point fourth-order finite-difference methods.

As far as storage is concerned — one stores either a block-tridiagonal or a tridiagonal matrix. Due to their simple form [the blocks are (3 × 3)] their solution never requires more than 10% of the total integration time — a cost that is offset by the advantage of increased accuracy — and by the implicit alternating direction implicit-splitting which allows a larger time-step. The method has been used by Cohn and others (1985) for the 2-D global barotropic primitive equations and has been proven by Chang and Shirer (1985) to provide the most accurate representation of the wave number distribution for the vorticity advection where the Arakawa (1966) Jacobian is used.

COMPUTER IMPLEMENTATION

Main program

The main program SHALL4 reads the first data card and after some preliminary calculation calls the subroutines SETUP, LOOK, HOUT, and UVOUT which display the initial height and velocity fields. The program then loops each time-step on the main subroutine ADIC4 which is concerned with the bulk of the ADI compact fourth-order algorithm and in turn calls subroutines BLKTRI and CYCTRI. These subroutines solve the block-tridiagonal or cyclic block-tridiagonal systems of linear equations resulting from the compact fourth-order ADI-discretization of the shallow-water equations.

After a predetermined number of time-steps subroutine LOOK is called, which calculates the three integral invariants of total mass, total energy and potential enstrophy. LOOK in turn calls subroutines HOUT and MAPPA. These subroutines perform the printing and the line-printer plotting of the height field, respectively. Subroutines SETUP, LOOK, MAPPA, HOUT, and UVOUT have been described in Navon (1979a).

Subroutine BLKTRI

In this subroutine, which is a specific block-tridiagonal solver optimized for the situation when the individual blocks are 3 × 3, a direct solution method is employed, based on clock — Gaussian elimination without pivoting.

Subroutine CYCTRI

This is a cyclic block-tridiagonal solver, written in such a way as also to take full advantage of the fact that the blocks in the coefficient matrices are 3 × 3. The algorithm used (see Navon; 1977) is a generalization to block-cyclic tridiagonal systems of the algorithm given by Ahlberg, Nilson, and Walsh (1967).

Subroutine ADIC4

This subroutine performs the bulk of the work involved in applying the compact fourth-order ADI algorithm, which has been described, including the implementation of the Adam $O(h^3)$ boundary conditions.

The subroutine exploits the (3 × 3) block matrices structure and in this sense is optimal computationally.

At the end of this routine, subroutine SMOOTH is called to filter out short-wave noise from the u, v, and h fields and thus prevent nonlinear aliasing.

User options

The user can determine the frequency of application of the smoother subroutine SMOOTH.

The printout enables the user, at a desired number of time-steps, to inspect the numerical and graphical display of the height field, the integral invariants of mass, total energy and potential enstrophy, and the CPU time used during the constant number of time-steps.

Input specification

The input to the program consists of two data cards, as follows.

CARD 1: FORMAT(6E10.4,I5) which contains the following seven 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 hour many time-steps the forecast fields and integral invariants of the shallow-water equations should be displayed.

CARD 2: (called in subroutine SETUP) specifies different parameters relative to the initial height field [Eqn (12)] using format 5E10.4, and contains the following five parameters.

HO	— 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.

Examples of output

Examples of the SHALL4 output are provided so as to demonstrate the different options of the program. The initial height field using a resolution of $\Delta x = \Delta y$ 200 km is shown in Figure 1, whereas Figure 2 shows the height-field contours after two days of simulation using a time-step of $\Delta t = 600$ s, also the different integral invariants of the shallow-water equations. Figure 3 shows the height-field contours after two days of integra-

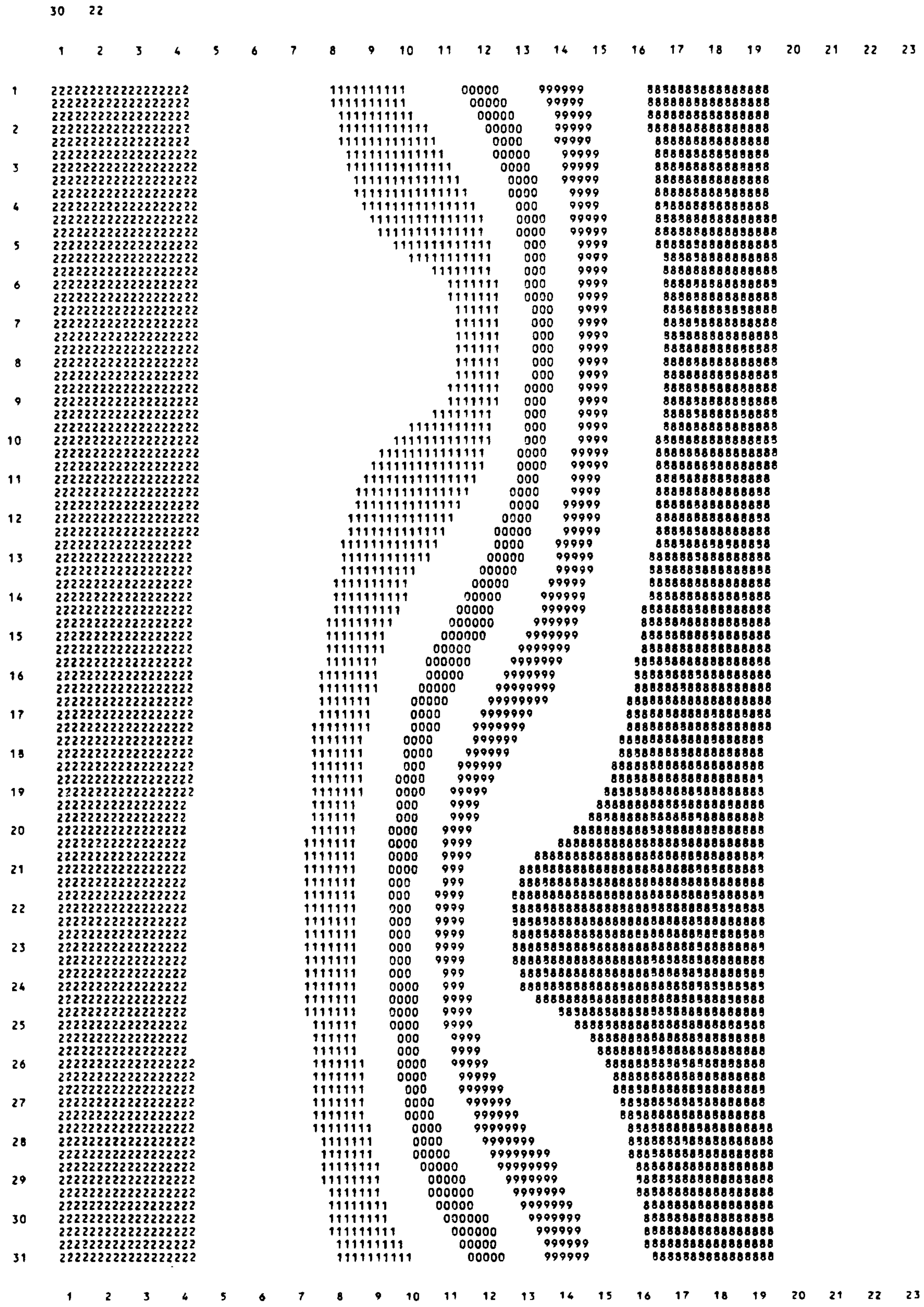


Figure 1. Initial height field.

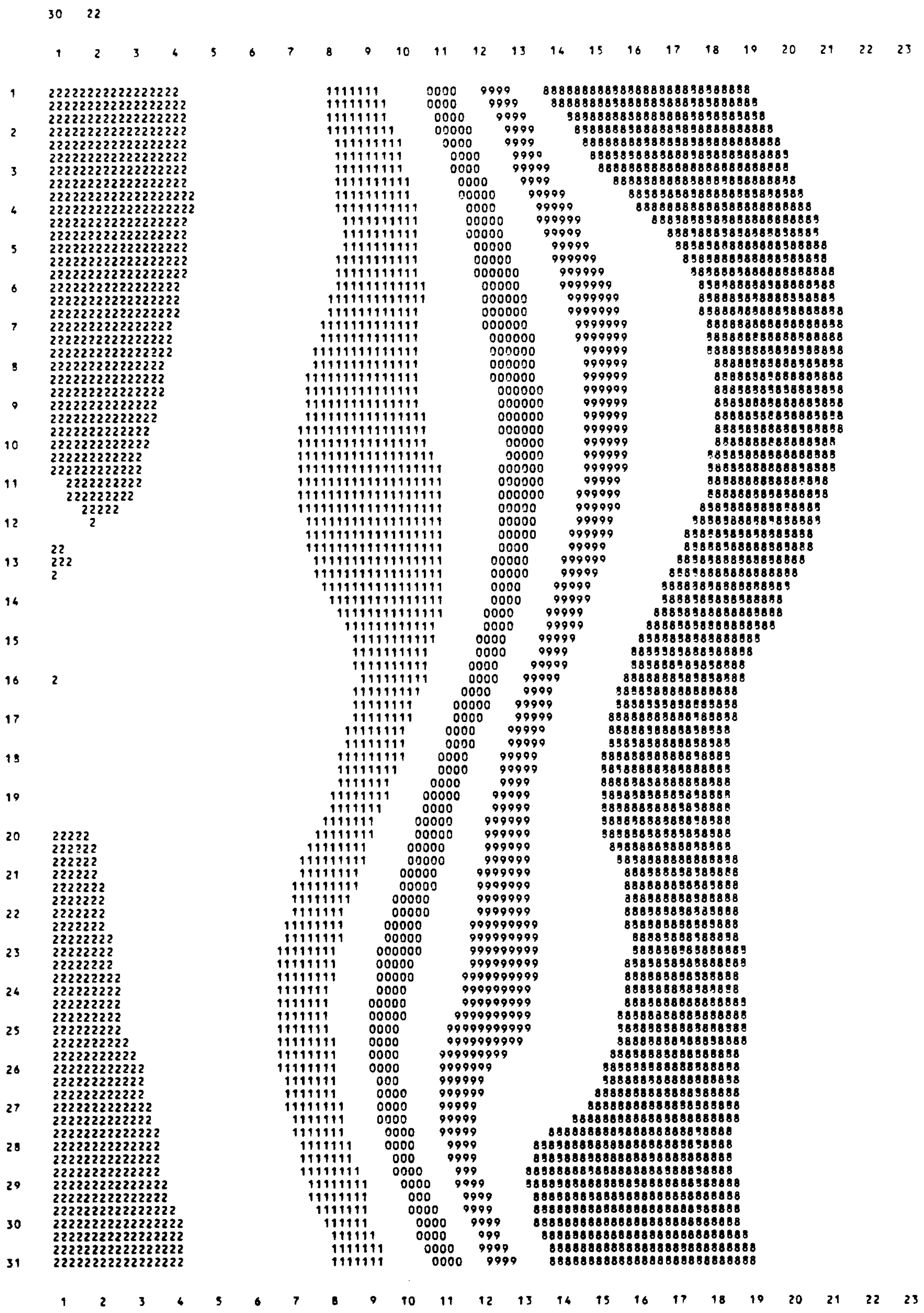
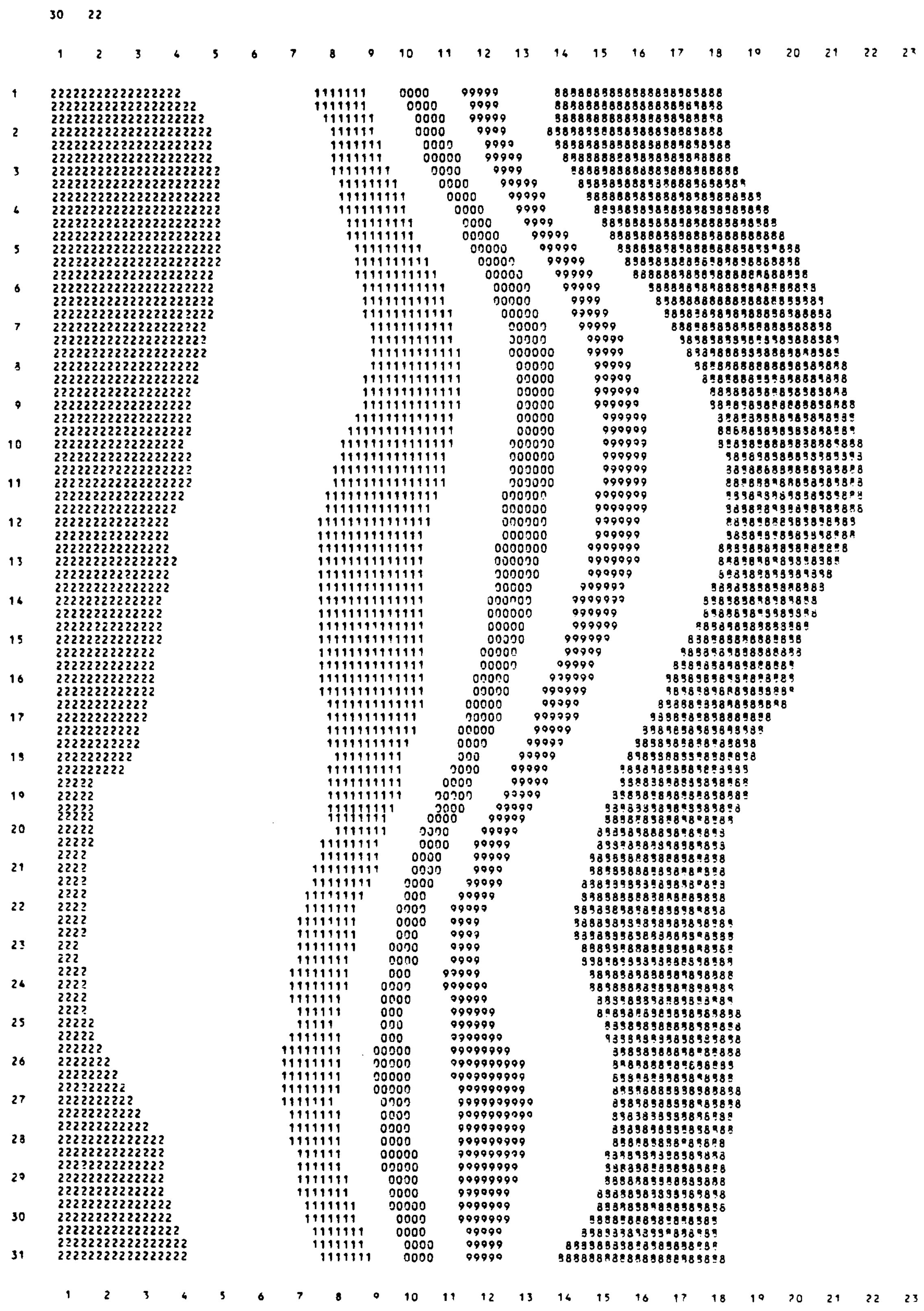


Figure 2. Height-field contours after two days of simulation using time-step Δ*t*-600 s.



```
LOOK

TIMESTEP 361    TIMESTEP 365    TIMESTEP 369

TIMESTEP 362    TIMESTEP 366    TIMESTEP 370

TIMESTEP 363    TIMESTEP 367    TIMESTEP 371

TIMESTEP 364    TIMESTEP 368    TIMESTEP 372
```

Figure 3. Height-field contours after two days of integration using time-step of 450 s.

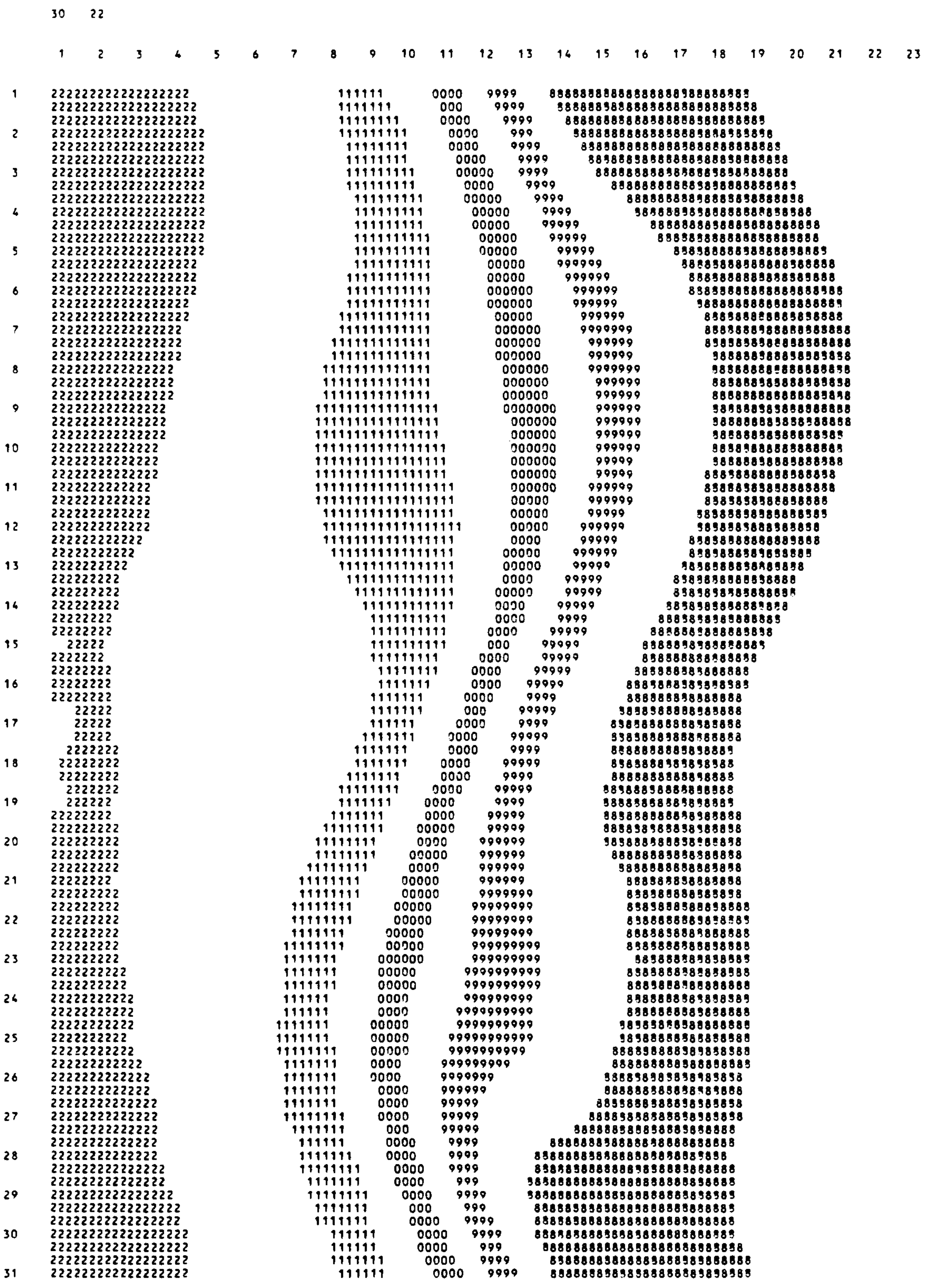


Figure 4. Height-field contours using time-step of 450 s.

tion using a time-step of 450 s, whereas Figure 4 shows the height-field contours using a time-step of 450 s, when the Shuman low-pass filter is applied every three time-steps.

REFERENCES

Adam, J., 1977, Highly-accurate compact implicit methods and boundary conditions: Jour. Comput. Physics, v. 24, p. 10–22.

Ahlberg, H. H., Nielson, E. N., and Walsh, J. L., 1967, The theory of splines and their application in mathematics in science and engineering: Academic Press, New York, 289 p.

Arakawa, A., 1966, Computational design for long-term numerical integration of the equations of fluid motion — the two dimensional incompressible flow: Part 1: Jour. Comput. Physics, v. 1, no. 1, p. 119–143.

Bates, J. R., 1984, The efficient semi-lagrangian and alternating direction implicit method for integrating the shallow-water equations: Mon. Wea. Rev., v. 112, no. 10, p. 2033–2047.

Beam, R. M., and Warming, R. F., 1976, An implicit finite-difference algorithm for hyperbolic systems in conservation-law form: Jour. Comput. Physics, v. 22, no. 1, p. 87–110.

Briley, W. R., and McDonald, H., 1977, Solution of the multi-dimensional compressible Navier-Stokes equations by a generalized implicit method: Jour. Comput. Physics, v. 24, no. 4, p. 372–397.

Chang, H. R., and Shirer, N. H., 1985, Compact spatial differencing techniques in numerical modeling: Mon. Wea. Rev., v. 113, no. 4, p. 409–423.

Climent, M., and Leventhal, S. H., 1975, Higher-order compact implicit schemes for the wave equation: Mathematics of Computation, v. 29, no. 132, p. 985–994.

Cohn, S. E., Dee, D., Isaacson, E., Marchesin, D., and Zwas, G., 1985, A fully implicit scheme for the shallow-water equations: Mon. Wea. Rev., v. 113, no. 4, p. 436–448.

Cullen, M. J. P., 1975, Application of the finite-element method to numerical weather prediction: unpubl. doctoral dissertation, Univ. Reading (UK), 90 p.

Cullen, M. J. P., 1977, The application of finite-element methods to the primitive equations of fluid motion, *in* Finite elements in water resources: Pentech Press, London, p. 4.231–4.236.

Cullen, M. J. P., and Morton, K. W., 1980, Analysis of evolutionary error in finite-element and other methods: Jour. Comput. Physics, v. 34, no. 2, p. 245–267.

Douglas, J., Jr., and Gunn, J. E., 1964, A general formulation of alternating-direction methods: Numer. Math., v. 6, no. 6, p. 428–453.

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

Grammeltvedt, A., 1969, A survey of finite-difference schemes for the primitive equations for a barotropic fluid: Mon. Wea. Rev., v. 97, no. 4, p. 384–404.

Gustafsson, B., 1975, The convergence rate for difference approximations to mixed initial boundary value problems: Mathematical Computing, v. 29, no. 130, p. 396–406.

Gustafsson, B., Kreiss, H. O., and Sundstrom, A., 1972, Difference approximations to mixed initial boundary value problems: Mathematical Computing, v. 26, no. 119, p. 649–686.

Haltiner, J. G., and Williams, R. T., 1980, Numerical prediction and dynamic Meteorology: John Wiley & Sons, New York, 477 p.

Hindmarsh, A. C., 1977, Solution of block-tridiagonal systems of linear algebraic equations: Lawrence Livermore Labs, P.O. Box 808, Livermore, California 94550, 25 p.

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.

Morton, K. W., 1977, Initial-value problems by finite-difference and other methods, *in* The state of art in numerical analysis: Academic Press, New York, p. 699–756.

Navon, I. M., 1977, Algorithms for solving scalar and block cyclic-tridiagonal systems of linear algebraic equations: CSIR Spec. Rept. WISK 265, Pretoria, South Africa, 29 p.

Navon, I. M., 1979a, ADIF, and FORTRAN IV program for solving the shallow-water equations: Computers & Geosciences, v. 5, no. 1, p. 19–39.

Navon, I. M., 1979b, Finite-element solution of the shallow-water equations on a limited area domain with three different mass-matrix formulations, *in* Fourth Conference on Numerical Weather Prediction: Silver-Spring, Maryland, p. 223–227.

Navon, I. M., 1983, A Numerov Galerkin technique applied to a finite-element shallow-water equations model with enforced conservation of integral invariants and selective limping: Jour. Comput. Physics, v. 52, no. 2, p. 313–339.

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

Shuman, F. G., 1957, Numerical methods in weather prediction II. smoothing and filtering: Mon. Wea. Rev., v. 85, no. 3, p. 357–361.

Steger, J. L., 1978, Coefficient matrices for implicit finite-difference solution of the inviscid fluid conservation-law equation: Computing Methods & Applications in Mechanical Engineering, v. 13, p. 175–188.

Temperton, C., 1975, Algorithms for the solution of cyclic tridiagonal systems: Jour. Comput. Physics, v. 19, no. 3, p. 317–323.

Takano, K., and Wurtele, M. G., 1982, A fourth-order energy and potential enstrophy conserving difference scheme: Air Force Geophysics Laboratory Report. AFGL-TR-82-0205, HANSCOM AFB, Massachusetts, 87 p.

Warming, R. F., and Beam, R. M., 1978, On the construction and application of implicit factored schemes for conservation-laws: Symposium in Applied Mathematics SIAM-AMS Proceedings, v. 11, Computational Fluid Dynamics, p. 85–129.

APPENDIX

```
C      PROGRAM SHALL4(INPUT,OUTPUT,TAPE1=INPUT,TAPE3=OUTPUT)
C
C       THE MAIN PROGRAM SHALL4 READS A DATA CARD AND CALLS SUBROUTINES
C       SETUP, LOOK AND UVOUT, PERFORMING INITIAL FIELD CALCULATIONS.
C
C       IT THEN CALLS SUBROUTINE ADIC4 WHICH PERFORMS THE COMPACT
C       FOURTH-ORDER CALCULATIONS.
C
C       EVERY IPR TIMESTEPS SUBROUTINE LOOK IS CALLED TO CALCULATE
C       VARIOUS DIAGNOSTICS.
C
       COMMON/CONST/FL,D,T,DX,DY,DT,G,TIME,IPR,IOPT
       COMMON/RITE/NIN,NOUT
       DIMENSION U(30,23),V(30,23),PHI(30,23),H(30,23),
      1DI(9,23),EI(9,23),FI(9,23),GI(3,23),
     2DJ(9,30),EJ(9,30),FJ(9,30),GJ(3,30),
     3X(3,30,23),Y(3,30,23),
     4A(4,30),B(4,30),F(23)
      DIMENSION TK(30,23),W(3,30,23),GA(9,30),TEM(30)
      G=10.
      NIN=1
      NOUT=3
      READ(NIN,10) FL,D,T,DX,DY,DT,IPR
   10 FORMAT(6E10.4,I5)
      LX=30
      LY=23
      NX=FL/DX
      NY=1+IFIX(D/DY)
      IF(NX.GT.LX) GO TO 30
      IF(NY.LE.LY) GO TO 50
   30 WRITE(NOUT,40)
   40 FORMAT(84HCHANGE ARRAY DIMENSIONS AND VALUES ASSIGNED TO LX AND LY
     1TO ACCOMMODATE THIS DATA SET)
      CALL EXIT
   50 CONTINUE
      NT=T/DT
      IF(IPR.EQ.0) IPR=1
      CALL SETUP(U,V,PHI,H,F,NX,NY,A,B,LX)
      TIME=0.
      CALL LOOK(U,V,PHI,H,NX,NY,LX,F)
      WRITE(NOUT,60)
   60 FORMAT(20H1    INITIAL U-FIELD)
      CALL UVOUT(U,NX,NY,LX)
      WRITE(NOUT,70)
   70 FORMAT(20H1    INITIAL V-FIELD)
      CALL UVOUT(V,NX,NY,LX)
      IOPT=0
C  LOOP FOR EACH TIMESTEP
      DO 90 IT=1,NT
      WRITE(NOUT,71) IT
   71 FORMAT(9H0TIMESTEP,I4)
      IOPT=IOPT+1
      CALL ADIC4(U,V,PHI,H,F,NX,NY,LX,
     1DI,EI,FI,GI,DJ,EJ,FJ,GJ,W,X,Y,GA,TK,TEM)
      TIME=TIME+DT
      IF(IOPT.GE.IPR) GO TO 80
      IF(IT.LT.NT) GO TO 90
  80  CALL LOOK(U,V,PHI,H,NX,NY,LX,F)
      IOPT=0
   90 CONTINUE
C  END OF TIME LOOP
      WRITE(NOUT,100)
  100 FORMAT(17H1    FINAL U-FIELD)
      CALL UVOUT(U,NX,NY,LX)
      WRITE(NOUT,110)
  110 FORMAT(17H1    FINAL V-FIELD)
      CALL UVOUT(V,NX,NY,LX)
      STOP
      END


      SUBROUTINE SETUP(U,V,PHI,H,F,NX,NY,S,C,LX)
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)
C
      COMMON/CONST/FL,D,T,DX,DY,DT,G,TIME,IPR,IOPT
      COMMON/RITE/NIN,NOUT
      DIMENSION U(LX,NY),V(LX,NY),PHI(LX,NY),F(NY),S(LX),C(LX),H(LX,NY)
      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=,E9.2,4H/SEC,
     1 12X,2HL=,F9.0,2H M,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
      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
      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,NX,NY,LX,F)
C
C      THIS SUBROUTINE CALCULATES THE TOTAL ENERGY, THE TOTAL MASS AND
C      POTENTIAL ENSTROPHY, WHICH ARE INTEGRAL INVARIANTS OF THE
C      SHALLOW WATER EQUATIONS.
C
C      IT ALSO PRINTS THE VALUES OF THE HEIGHT FIELD BY CALLING
C      SUBROUTINE HOUT.
C
      REAL MSVRT
      COMMON/N/ NAME
      COMMON/CONST/FL,D,T,DX,DY,DT,G,TIME,IPR,IOPT
      COMMON/RITE/NIN,NOUT
      DIMENSION U(LX,NY),V(LX,NY),PHI(LX,NY),H(LX,NY),F(NY)
      DATA IND/0/,NSTEP/0/,TIMEA/0./
    2 FORMAT(1H1)
    3 FORMAT(7H1 TIME=,F9.0,4H SEC,10X,6HHMEAN=,F8.2,2H M,10X,7HENERGY=,
     1 1PE12.6,10X,12HCPU TIME FOR,I4,8H STEPS =,0PF8.2,4H SEC)
    4 FORMAT(23H MEAN SQUARE VORTICITY=,1PE13.6)
   55 FORMAT(5X,4HLOOK)
      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
      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
      MSVRT=0.
      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 CONTINUE
      WRITE(NOUT,55)
   50 RETURN
      END

      SUBROUTINE MAPPA(FUN,C,NX,NZ,LX)
C
C     THIS SUBROUTINE PROVIDES A VISUAL DISPLAY OF THE FIELD BY
C     PRINTING AN ISOLINE CONTOUR OF THE FIELD USING THE DIGITS 0 TO 9.
C
C     THE PARAMETER FUN GIVES THE FIELD TO BE CONTOURED, WHILE C IS A
C     PARAMETER GIVING THE INVERSE OF THE CONTOUR CONSTANT.
C
      COMMON/RITE/ NIN,NOUT
      DIMENSION FUN(LX,NZ) ,ANS(4,116),IANS(116),NUM(10)
      DATA NUM(1)/1H1/,NUM(2)/1H2/,NUM(3)/1H3/,NUM(4)/1H4/,NUM(5)/1H5/,
     1     NUM(6)/1H6/,NUM(7)/1H7/,NUM(8)/1H8/,NUM(9)/1H9/,NUM(10)/1H0/,
     2     BLNK/1H /
  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
      WRITE(NOUT,111) NX,NY
      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)
      JXPN=JX+N
      DO 20 L=1,LEND
      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
C      THIS SUBROUTINE PRINTS OUT THE VALUES OF THE VELOCITY FIELD
C      COMPONENTS IN MATRIX FORM.
C
C      W STANDS FOR EITHER U OR V COMPONENTS OF THE VELOCITY FIELD.
C
      COMMON/RITE/NIN,NOUT
      DIMENSION W(LX,NY)
    1 FORMAT(3HO  ,12I11/)
      JE=0
    2 FORMAT(1X,I2,12E11.4)
 5      JB=JE+1
      JE=MINO(NX,JE+12)
      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
C      THIS SUBROUTINE PRINTS OUT THE HEIGHT FIELD VALUES IN MATRIX FORMAT.
C
      COMMON/RITE/NIN,NOUT
      DIMENSION H(LX,NY)
    6 FORMAT(15HO HEIGHT VALUES/)
    7 FORMAT(3X,22I6/)
    8 FORMAT(1X,I2,22F6.0)
      JE=0
 5    JB=JE+1
      JE=MINO(NX,JE+22)
      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
       END

      SUBROUTINE ADIC4(U,V,PHI,H,F,NX,NY,LX,
     1DI,EI,FI,GI,DJ,EJ,FJ,GJ,W,X,Y,GA,TK,TEM)
C
C      THIS SUBROUTINE PERFORMS THE FOURTH-ORDER COMPACT SOLUTION OF THE
C      SHALLOW WATER EQUATIONS.
C      THE SUBROUTINE EXPLOITS THE (3*3) BLOCK-MATRICES STRUCTURES FOR
C      COMPUTATIONAL ECONOMY.
C      IT USES SUBROUTINE BLKTRI AND CYCTRI FOR BLOCK AND CYCLIC BLOCK
C      TRIDIAGONAL SOLUTION OF THE LINEAR ALGEBRAIC EQUATIONS SYSTEMS.
C
      COMMON/CONST/FL,D,T,DX,DY,DT,G,TIME,IPR,IOPT
      DIMENSION U(LX,NY),V(LX,NY),PHI(LX,NY),H(LX,NY),F(NY),
     1DI(9,NY),EI(9,NY),FI(9,NY),GI(3,NY),
     2DJ(9,NX),EJ(9,NX),FJ(9,NX),GJ(3,NX),
     3W(3,LX,NY),X(3,LX,NY),Y(3,LX,NY),GA(9,NX),TK(NX,NY),TEM(LX)
      NX1=NX-1
      NX2=NX-2
      NY1=NY-1
      NY2=NY-2
      T1=.5*DT
      C3=1./DY
      C1=.5*C3
      C2=3.*C3
      C4=2.*C3
      A1=1.5*DT/DX
      A3=A1/3.
      C=DT*DT*DT
      C5=C/(DX*DX)
      C6=C/(DY*DY)
      EPS=1.
C
C     (A)
C
      DO 5 K=1,9
      DO 5 J=1,NY
      DI(K,J)=0.
      EI(K,J)=0.
    5 FI(K,J)=0.
      DO 10 K=1,9,4
      EI(K,1)=1.
      FI(K,1)=2.
      DI(K,NY)=2.
      EI(K,NY)=1.
      EI(K,2)=2.
      FI(K,2)=1.
      DI(K,NY1)=1.
      EI(K,NY1)=2.
      DO 10 J=3,NY2
      DI(K,J)=1.
      EI(K,J)=4.
   10 FI(K,J)=1.
      DO 30 I=1,NX
      DO 15 J=1,NY
      S=H(I,J)*V(I,J)
      Y(1,I,J)=S*U(I,J)
      Y(2,I,J)=S*V(I,J)+G*H(I,J)*H(I,J)
   15 Y(3,I,J)=S
      DO 20 K=1,3
      GI(K,1)=C1*(-5.*Y(K,I,1)+4.*Y(K,I,2)+Y(K,I,3))
      GI(K,  2)=C1*( 5.*Y(K,I,  3)-4.*Y(K,I,  2)-Y(K,I,  1))
      GI(K,NY1)=C1*(-5.*Y(K,I,NY2)+4.*Y(K,I,NY1)+Y(K,I, NY))
      GI(K,NY)=C1*(5.*Y(K,I,NY)-4.*Y(K,I,NY-1)-Y(K,I,NY-2))
      DO 20 J=3,NY2
   20 GI(K,J)=C2*(Y(K,I,J+1)-Y(K,I,J-1))
C
      CALL BLKTRI(DI,EI,FI,GI,NY,1,NY,GA)
C
      DO 25 J=1,NY
      S=H(I,J)*V(I,J)
      X(1,I,J)=H(I,J)*U(I,J)+T1*(S*F(J)+GI(1,J))
      X(2,I,J)=S+T1*GI(2,J)
      X(3,I,J)=H(I,J)+T1*GI(3,J)
   25 CONTINUE
      X(2,I,1)=0.
      X(2,I,NY)=0.
   30 CONTINUE
C
C     (B)
C
      DO 35 K=1,9
      DO 35 J=1,NY
      DI(K,J)=0.
      EI(K,J)=0.
   35 FI(K,J)=0.
      DO 37 K=1,9,4
      EI(K,1)=1.
      FI(K,1)=2.
      EI(K,2)=2.
      FI(K,2)=1.
      DI(K,NY1)=1.
      EI(K,NY1)=2.
      DI(K,NY)=2.
      EI(K,NY)=1.
      DO 37 J=3,NY2
      DI(K,J)=1.
      EI(K,J)=4.
   37 FI(K,J)=1.
      DO 55 I=1,NX
      DO 40 J=1,NY
 40   Y(2,I,J)=Y(2,I,J)-.5*G*H(I,J)*H(I,J)
      DO 45 K=1,3
      GI(K,1)=C1*(-5.*Y(K,I,1)+4.*Y(K,I,2)+Y(K,I,3))
      GI(K,  2)=C1*( 5.*Y(K,I,  3)-4.*Y(K,I,  2)-Y(K,I,  1))
      GI(K,NY1)=C1*(-5.*Y(K,I,NY2)+4.*Y(K,I,NY1)+Y(K,I, NY))
      GI(K, NY)=C1*( 5.*Y(K,I, NY)-4.*Y(K,I,NY1)-Y(K,I,NY2))
      DO 45 J=3,NY2
   45 GI(K,J)=C2*(Y(K,I,J+1)-Y(K,I,J-1))
C
      CALL BLKTRI(DI,EI,FI,GI,NY,1,NY,GA)
C
      DO 50 J=1,NY
      S=X(3,I,J)+2.*H(I,J)
      R=X(3,I,J)-H(I,J)
      Y(1,I,J)=2.*U(I,J)*X(1,I,J)-U(I,J)*U(I,J)*S+G*H(I,J)*R
      Y(2,I,J)=V(I,J)*X(1,I,J)+U(I,J)*X(2,I,J)-U(I,J)*V(I,J)*S
      Y(3,I,J)=X(1,I,J)-2.*H(I,J)*U(I,J)
      X(1,I,J)=X(1,I,J)-DT*GI(1,J)
      X(2,I,J)=X(2,I,J)-DT*GI(2,J)-T1*F(J)*X(1,I,J)
      X(3,I,J)=X(3,I,J)-DT*GI(3,J)
   50 CONTINUE
   55 CONTINUE
C
C     (C)
C
      DO 65 I=1,NX
      DJ(2,I)=0.
      DJ(7,I)=-A1
      DJ(8,I)=0.
      DJ(9,I)=1.
      FJ(2,I)=0.
      FJ(7,I)=A1
      FJ(8,I)=0.
      FJ(9,I)=1.
      DO 60 K=1,9
   60 EJ(K,I)=0.
      DO 65 K=1,9,4
   65 EJ(K,I)=4.
      DO 95 J=1,NY
      DO 70 K=1,3
      GJ(K,1)=X(K,NX,J)+4.*X(K,1,J)+X(K,2,J)+A1*(Y(K,2,J)-Y(K,NX,J))
      GJ(K,NX)=X(K,NX-1,J)+4.*X(K,NX,J)+X(K,1,J)+A1*(Y(K,1,J)-Y(K,NX-1,J
     1))
      DO 70 I=2,NX1
   70 GJ(K,I)=X(K,I-1,J)+4.*X(K,I,J)+X(K,I+1,J)+A1*(Y(K,I+1,J)-Y(K,I-1,J
     1))
      S=A1*U(NX,J)
      DJ(1,1)=1.-2.*S
      DJ(3,1)=S*U(NX,J)-A1*G*H(NX,J)
      DJ(4,1)=-A1*V(NX,J)+T1*F(J)
      DJ(5,1)=1.-S
      DJ(6,1)=S*V(NX,J)
      S=A1*U(1,J)
      FJ(1,NX)=1.+2.*S
      FJ(3,NX)=-S*U(1,J)+A1*G*H(1,J)
      FJ(4,NX)=A1*V(1,J)+T1*F(J)
      FJ(5,NX)=1.+S
      FJ(6,NX)=-S*V(1,J)
      DO 75 I=1,NX
   75 EJ(4,I)=4.*T1*F(J)
      DO 80 I=2,NX
      S=A1*U(I-1,J)
      DJ(1,I)=1.-2.*S
      DJ(3,I)=S*U(I-1,J)-A1*G*H(I-1,J)
      DJ(4,I)=-A1*V(I-1,J)+T1*F(J)
      DJ(5,I)=1.-S
   80 DJ(6,I)=S*V(I-1,J)
      DO 85 I=1,NX1
      S=A1*U(I+1,J)
      FJ(1,I)=1.+2.*S
      FJ(3,I)=-S*U(I+1,J)+A1*G*H(I+1,J)
      FJ(4,I)=A1*V(I+1,J)+T1*F(J)
      FJ(5,I)=1.+S
   85 FJ(6,I)=-S*V(I+1,J)
C
      CALL CYCTRI(DJ,EJ,FJ,GJ,NX,W,GA)
C
      DO 90 K=1,3
      DO 90 I=1,NX
   90   X(K,I,J)=GJ(K,I)
   95 CONTINUE
      DO 91 I=1,NX
      X(2,I,1)=0.
      X(2,I,NY)=0.
   91 CONTINUE
C
C     (D)
C
      DO 100 K=1,9
      DI(K,1)=0.
  100 FI(K,NY)=0.
      EI(4,1)=0.
      EI(7,1)=0.
      EI(8,1)=-A3
      EI(9,1)=1.
      FI(4,1)=0.
      FI(7,1)=0.
      FI(8,1)=A3
      FI(9,1)=0.
      DI(4,NY)=0.
      DI(7,NY)=0.
      DI(8,NY)=-A3
      DI(9,NY)=0.
      EI(4,NY)=0.
      EI(7,NY)=0.
      EI(8,NY)=A3
      EI(9,NY)=1.
      DO 115 J=2,NY1
      DI(4,J)=0.
      DI(7,J)=0.
      DI(8,J)=-A1
      DI(9,J)=1.
      DO 105 K=1,9
  105 EI(K,J)=0.
      DO 110 K=1,9,4
  110 EI(K,J)=4.
      EI(2,J)=-4.*T1*F(J)
      FI(4,J)=0.
      FI(7,J)=0.
      FI(8,J)=A1
  115 FI(9,J)=1.
      DO 135 I=1,NX
      DO 120 K=1,3
      GI(K,1)=  X(K,I,1)
      GI(K,NY)=  X(K,I,NY)
      DO 120 J=2,NY1
  120 GI(K,J)=  X(K,I,J-1)+4.*  X(K,I,J)+  X(K,I,J+1)
      S=A3*V(I,1)
      R=A3*U(I,1)
      EI(1,1)=1.-S
      EI(2,1)=-R-T1*F(1)
      EI(3,1)=R*V(I,1)
      EI(5,1)=1.-2.*S
      EI(6,1)=S*V(I,1)-A3*G*H(I,1)
      S=A3*V(I,2)
      R=A3*U(I,2)
      FI(1,1)=S
      FI(2,1)=R
      FI(3,1)=-R*V(I,2)
      FI(5,1)=2.*S
      FI(6,1)=-S*V(I,2)+A3*G*H(I,2)
      S=A3*V(I,NY-1)
      R=A3*U(I,NY-1)
      DI(1,NY)=-S
      DI(2,NY)=-R
      DI(3,NY)=R*V(I,NY-1)
      DI(5,NY)=-2.*S
      DI(6,NY)=S*V(I,NY-1)-A3*G*H(I,NY-1)
      S=A3*V(I,NY)
      R=A3*U(I,NY)
      EI(1,NY)=1.+S
      EI(2,NY)=R-T1*F(NY)
      EI(3,NY)=-R*V(I,NY)
      EI(5,NY)=1.+2.*S
      EI(6,NY)=-S*V(I,NY)+A3*G*H(I,NY)
      DO 125 J=2,NY1
      S=A1*V(I,J-1)
      R=A1*U(I,J-1)
      DI(1,J)=1.-S
      DI(2,J)=-R-T1*F(J-1)
      DI(3,J)=R*V(I,J-1)
      DI(5,J)=1.-2.*S
      DI(6,J)=S*V(I,J-1)-A1*G*H(I,J-1)
      S=A1*V(I,J+1)
      R=A1*U(I,J+1)
      FI(1,J)=1.+S
      FI(2,J)=R-T1*F(J+1)
      FI(3,J)=-R*V(I,J+1)
      FI(5,J)=1.+2.*S
  125 FI(6,J)=-S*V(I,J+1)+A1*G*H(I,J+1)
C
      CALL BLKTRI(DI,EI,FI,GI,NY,1,NY,GA)
C
      DO 130 J=1,NY
      S=1./GI(3,J)
      U(I,J)=GI(1,J)*S
      V(I,J)=GI(2,J)*S
      H(I,J)=GI(3,J)
      PHI(I,J)=2.*SQRT(G*H(I,J))
  130 CONTINUE
      V(I,1)=0.
      V(I,NY)=0.
  135 CONTINUE
      IIOPT=2*(IOPT/2)
      IF(IIOPT.NE.IOPT) GO TO 137
      CALL SMOOTH(U,TEM,LX,NY)
      CALL SMOOTH(V,TEM,LX,NY)
      CALL SMOOTH(H,TEM,LX,NY)
  137 CONTINUE
      RETURN
      END

      SUBROUTINE SMOOTH(ZK,TEM,LX,NY)
C
C      THIS SUBROUTINE FILTERS OUT SHORT WAVES FROM THE U, V, AND H FIELDS
C      USING THE SCHUMAN FILTER TO PREVENT NONLINEAR ALIASING.
C      THE FILTER ACTS FIRST IN THE Y- AND THEN IN THE X-DIRECTION.
C
      DIMENSION C(3,2),ZK(LX,NY,1),TEM(LX)
      DATA C(1,1)/3.8798/,C(2,1)/-1.77097/,C(3,1)/0.331065/,
     1     C(1,2)/0.375/,C(2,2)/0.25/,C(3,2)/0.0625/
       NX=12
      NX1=NX-1
      NY1=NY-1
      M=1
      DO 300 KK=1,2
      C1=C(1,KK)
      C2=C(2,KK)
      C3=C(3,KK)
C     SMOOTH IN Y-DIRECTION
      DO 301 I=1,NX
      DO 3005 J=2,NY1
      IF(J.EQ.2) GO TO 25
      IF(J.EQ.NY1) GO TO 35
      GO TO 40
   25 TWO=ZK(I,J+2,M)+2.*ZK(I,J-1,M)-ZK(I,J,M)
      GO TO 50
   35 TWO=2.*ZK(I,J+1,M)-ZK(I,J,M)+ZK(I,J-2,M)
      GO TO 50
   40 TWO=ZK(I,J+2,M)+ZK(I,J-2,M)
   50 CONTINUE
      TEM(J)=C1*ZK(I,J,M)+C2*(ZK(I,J-1,M)+ZK(I,J+1,M))+TWO*C3
 3005 CONTINUE
      DO 3006 J=2,NY1
 3006 ZK(I,J,M)=TEM(J)
  301 CONTINUE
C     SMOOTH IN X-DIRECTION
      DO 303 J=1,NY
      DO 302 I=2,NX1
      IF(I.EQ.2) GO TO 170
      IF(I.EQ.NX1) GO TO 185
      GO TO 200
  170 TWO=2.*ZK(I-1,J,M)-ZK(I,J,M)+ZK(I+2,J,M)
      GO TO 230
  185 TWO=ZK(I-2,J,M)+2.*ZK(I+1,J,M)-ZK(I,J,M)
      GO TO 230
  200 TWO=ZK(I-2,J,M)+ZK(I+2,J,M)
  230 TEM(I)=C1*ZK(I,J,M)+C2*(ZK(I-1,J,M)+ZK(I+1,J,M))+C3*TWO
  302 CONTINUE
      DO 3025 I=2,NX1
 3025 ZK(I,J,M)=TEM(I)
  303 CONTINUE
  300 CONTINUE
      RETURN
      END

      SUBROUTINE BLKTRI(A,B,C,D,N,MM,NN,G)
C
C      THIS IS A BLOCK-TRIDIAGONAL SOLVER OPTIMIZED FOR THE CASE WHEN
C      THE INDIVIDUAL BLOCK MATRICES ARE (3*3).
C      A DIRECT SOLUTION BASED ON GAUSSIAN ELIMINATION I. USED.
C
      DIMENSION A(9,NN),B(9,NN),C(9,NN),D(3,NN,MM),E(9),V(9),G(9,NN)
       DO 110 I=1,N
      IM=I-1
C
C       ALPHA
C
       DO 10 K=1,9
 10    E(K)=B(K,I)
      IF(I.EQ.1) GO TO 30
       DO 20 K=1,9
      L=3*((K-1)/3)+1
       L1=L+1
       L2=L+2
       M=K-L+1
      M3=M+3
       M6=M+6
  20   E(K)=E(K)+A(L,I)*G(M,IM)+A(L1,I)*G(M3,IM)+A(L2,I)*G(M6,IM)
  30   CONTINUE
C
C     INVERSE OF ALPHA
C
       V(1)=E(5)*E(9)-E(6)*E(8)
       V(2)=E(3)*E(8)-E(2)*E(9)
       V(3)=E(2)*E(6)-E(3)*E(5)
       V(4)=E(6)*E(7)-E(4)*E(9)
       V(5)=E(1)*E(9)-E(3)*E(7)
       V(6)=E(3)*E(4)-E(1)*E(6)
       V(7)=E(4)*E(8)-E(5)*E(7)
       V(8)=E(2)*E(7)-E(1)*E(8)
       V(9)=E(1)*E(5)-E(2)*E(4)
       DET=1./(E(1)*V(1)+E(2)*V(4)+ E(3)*V(7))
        DO 40 K=1,9
 40     V(K)=DET*V(K)
C
C      GAMMA
C
      DO 50 K=1,9
       L=3*((K-1)/3)+1
       L1=L+1
       L2=L+2
       M=K-L+1
      M3=M+3
       M6=M+6
 50    E(K)=V(L)*C(M,I)+V(L1)*C(M3,I)+V(L2)*C(M6,I)
       DO 60 K=1,9
 60    G(K,I)=-E(K)
```

```
C
C        Y
C
       DO 110 J=1,MM
      DO 70 K=1,3
 70   E(K)=D(K,I,J)
       IF(I.EQ.1) GO TO 90
      L=1
       DO 80 K=1,3
      L1=L+1
      L2=L+2
      E(K)=E(K)-A(L,I)*D(1,IM,J)-A(L1,I)*D(2,IM,J)-A(L2,I)*D(3,IM,J)
 80   L=L+3
 90    CONTINUE
C
      L=1
      DO 100 K=1,3
      L1=L+1
      L2=L+2
       D(K,I,J)=V(L)*E(1)+V(L1)*E(2)+V(L2)*E(3)
 100     L=L+3
 110   CONTINUE
C
C        X
C
      M=N
      DO 140 I=2,N
      MP=M
      M=M-1
       DO 140 J=1,MM
      L=1
        DO 120 K=1,3
       L1=L+1
      L2=L+2
      E(K)=D(K,M,J)+G(L,M)*D(1,MP,J)+G(L1,M)*D(2,MP,J)+G(L2,M)*D(3,MP,J)
 120   L=L+3
       DO 130 K=1,3
 130  D(K,M,J)=E(K)
 140    CONTINUE
      RETURN
      END

      SUBROUTINE CYCTRI(P,Q,R,D,N,W,G)
C
C      THIS IS A CYCLIC BLOCK TRIDIAGONAL SOLVER USING THE FACT THAT THE
C      UNIT BLOCK MATRICES ARE (3*3).
C      THIS ALGORITHM GENERALIZES THE AHLBERG, NILSON AND WALSH METHOD (1967).
C
      DIMENSION P(9,N),Q(9,N),R(9,N),D(3,N),W(3,N,4),E(9),V(9),G(9,N)
C
      NM=N-1
      DO 10 I=1,NM
      DO 10 K=1,3
       W(K,I,1)=D(K,I)
       W(K,I,2)=0.
       W(K,I,3)=0.
 10    W(K,I,4)=0.
      W(1,1,2)=P(1,1)
       W(2,1,2)=P(4,1)
      W(3,1,2)=P(7,1)
      W(1,1,3)=P(2,1)
       W(2,1,3)=P(5,1)
       W(3,1,3)=P(8,1)
       W(1,1,4)=P(3,1)
       W(2,1,4)=P(6,1)
       W(3,1,4)=P(9,1)
       W(1,NM,2)=R(1,NM)
       W(2,NM,2)=R(4,NM)
       W(3,NM,2)=R(7,NM)
       W(1,NM,3)=R(2,NM)
       W(2,NM,3)=R(5,NM)
       W(3,NM,3)=R(8,NM)
       W(1,NM,4)=R(3,NM)
       W(2,NM,4)=R(6,NM)
       W(3,NM,4)=R(9,NM)
C
      CALL BLKTRI(P,Q,R,W,NM,4,N,G)
C
      DO 20 K=1,9
      L=3*((K-1)/3)+1
      L1=L+1
      L2=L+2
      M=K-L+2
 20    E(K)=Q(K,N)-R(L,N)*W(1,1,M)-R(L1,N)*W(2,1,M)-R(L2,N)*W(3,1,M)
     1          -P(L,N)*W(1,NM,M)-P(L1,N)*W(2,NM,M)-P(L2,N)*W(3,NM,M)
C
      V(1)=E(5)*E(9)-E(6)*E(8)
      V(2)=E(3)*E(8)-E(2)*E(9)
      V(3)=E(2)*E(6)-E(3)*E(5)
      V(4)=E(6)*E(7)-E(4)*E(9)
      V(5)=E(1)*E(9)-E(3)*E(7)
      V(6)=E(3)*E(4)-E(1)*E(6)
      V(7)=E(4)*E(8)-E(5)*E(7)
      V(8)=E(2)*E(7)-E(1)*E(8)
      V(9)=E(1)*E(5)-E(2)*E(4)
      DET=1./(E(1)*V(1)+E(2)*V(4)+E(3)*V(7))
      DO 30 K=1,9
 30   V(K)=DET*V(K)
C
      L=1
      DO 40 K=1,3
      L1=L+1
       L2=L+2
      E(K)=D(K,N)-R(L,N)*W(1,1,1)-R(L1,N)*W(2,1,1)-R(L2,N)*W(3,1,1)
     1           -P(L,N)*W(1,NM,1)-P(L1,N)*W(2,NM,1)-P(L2,N)*W(3,NM,1)
 40   L=L+3
C
      D1N=V(1)*E(1)+V(2)*E(2)+V(3)*E(3)
      D2N=V(4)*E(1)+V(5)*E(2)+V(6)*E(3)
      D3N=V(7)*E(1)+V(8)*E(2)+V(9)*E(3)
C
      DO 50 K=1,3
       DO 50 I=1,NM
      D(K,I)=W(K,I,1)-W(K,I,2)*D1N-W(K,I,3)*D2N-W(K,I,4)*D3N
 50   CONTINUE
C
      D(1,N)=D1N
      D(2,N)=D2N
      D(3,N)=D3N
C
       RETURN
      END
```